Warsaw University of Technology

DISCIPLINE OF SCIENCE INFORMATION AND COMMUNICATION TECHNOLOGY

FIELD OF SCIENCE ENGINEERING AND TECHNOLOGY

Ph.D. Thesis

Kamil Deja, M.Sc.

Data representations in generative modelling

Supervisor

Tomasz Trzciński, PhD DSc

WARSAW 2023

Acknowledgements

First of all I would like to express my sincere gratitude to my supervisor Tomasz Trzciński. Thank you for taking me onboard for this fascinating journey we have shared over the past few years. Your guidance and support have been invaluable throughout my scientific career, and I am grateful for the opportunity to explore the exciting side of research under your supervision.

I would also like to extend my thanks to the brilliant individuals I had the pleasure of working with. Many thanks to Paweł Wawrzyński for our stimulating discussions, valuable comments and out-of-the box ideas. Sincere appreciation to Jakub M. Tomczak, who warmly hosted me in Amsterdam while showing me how much I still have to learn about generative modelling.

My heartfelt thanks go to my coauthors, for their hard work in our joint projects – among the others: Wojciech Masarczyk, Daniel Marczak, Jan Dubiński, Anna Kuzina, Georgi Tinchev and Ariadna Sanchez. Sincere appreciation to my other colleagues from Warsaw, Amsterdam and Amazon, especially: Monika Wysoczańska, Łukasz Neumann, Kacper Kania, Marta Czarnowska. Thank you all for great support in this journey. Lastly, I extend a special thanks to Mateusz Klimaszewski, who has proven to be the finest interactive rubber duck I could have ever envisioned.

I am immensely grateful to my wife and family for their unwavering support, which has been instrumental in enabling me to pursue my scientific career. Thank you for giving me a foundation to pursue my scientific career, and thank you for instilling within me a deep-seated passion for continuous learning.

Data representations in generative modelling

This thesis presents a series of publications contributing to an in-depth understanding and development of generative models. We specifically focus on two recent approaches: generative autoencoders and diffusion-based generative models. We analyse how these methods build internal data representations and how they change when a model is retrained with additional data. We also propose several novel methods for generative modelling and their extensions to a continual learning setup.

In the first part of our work, we provide an overview of different generative autoencoders. We then introduce a novel model which allows for flexible encoding of examples into data representations, leveraging an additional neural network for sampling new data points.

In the second part, we move on to the recently proposed diffusion-based generative models. We begin by presenting an in-depth analysis of how the intermediate representations of images change with diffusion timesteps. Next, we introduce a novel joint model that demonstrates how the data representations generated by a diffusion model can be utilised to enhance performance in downstream tasks.

Finally, we extend our analysis to a continual learning scenario. Here, we show that generative approaches can be used as a universal method for continuous knowledge accumulation within models. To that end, we introduce two methods for continual generative modelling. In the first one, we propose a binary autoencoder that efficiently stores past experiences, while the second one is a method for the continuous alignment of data representations in the Variational Autoencoder's latent space.

Overall, our work contributes to the development of generative models through an in-depth analysis of their internal representations and novel ways of their application to real-life problems and continual-learning scenario.

Keywords: Generative Models, Continual Learning, Variational Autoencoder, Diffusion Models

Reprezentacje danych w modelowaniu generatywnym

W niniejszej pracy przedstawiamy serię publikacji poświęconą analizom i oryginalnym metodom modelowania generatywnego. W szczególności skupiamy się na sposobie budowania reprezentacji danych za pomocą powyższych metod. Badamy też jak wewnętrzne reprezentacje zmieniają się przy dotrenowywaniu modelu w oparciu o dodatkowe dane. W ramach przedstawionych prac proponujemy kilka nowych modeli generatywnych, wraz z ich rozszerzeniem do problemu uczenia ciągłego.

W pierwszej części pracy, dokonujemy przeglądu różnych autoenkoderów generatywnych. Następnie przedstawiamy nasz nowy model, który umożliwia elastyczne kodowanie przykładów do wewnętrznych reprezentacji, wykorzystując dodatkową sieć neuronową do próbkowania nowych obserwacji.

W drugiej części, przechodzimy do zaproponowanych niedawno modeli generatywnych działających na zasadzie procesu dyfuzji. W pierwszej kolejności przedstawiamy analizy tego jak tymczasowe reprezentacje danych zmieniają się wraz z krokami dyfuzji. Następnie wprowadzamy nowy model łączny, za pomocą którego pokazujemy jak reprezentacje danych tworzone w procesie dyfuzji mogą być wykorzystane do poprawy wydajności w różnych zadaniach.

Następnie, rozszerzamy naszą analizę na problem uczenia ciągłego. W zagadnieniu tym pokazujemy że modele generatywne mogą być używane jako uniwersalna metoda do gromadzenia wiedzy napływającej w porcjach. W szczególności, przedstawiamy nasze dwie autorskie metody. W pierwszej wprowadzamy binarny autoenkoder, który wykorzystujemy do efektywnego przechowywania przeszłych doświadczeń. Natomiast w drugiej pracy, pokazujemy jak wykorzystać wariacyjny autoenkoder do ciągłej konsolidacji wiedzy poprzez uspójnianie ukrytych reprezentacji danych.

Podsumowując, w niniejszej pracy prezentujemy dogłębną analizę wewnętrznych reprezentacji modeli generatywnych oraz ich nowatorskie zastosowania w realnych problemach, włączając w to uczenia ciągłego.

Sowa kluczowe: Modele Generatywne, Uczenia Ciągłe, Autoenkodery Wariacyjne, Modele Dyfuzyjne

Contents

1. Introduction

Generative models gain increasing attention due to the number of remarkable applications, where methods such as Variational Autoencoders (VAE) (Kingma and Welling, 2014), Generative Adversarial Networks (GAN) (Goodfellow et al., 2014a) or Diffusion-Based Generative Models (DDGMs) (Sohl-Dickstein et al., 2015) play an important role. The most common ones include synthetic images generation (Brock et al., 2018; Dhariwal and Nichol, 2021; Song et al., 2020a), with the extension to the methods creating new images from textual prompts such as DALLE2 (Ramesh et al., 2022) or Imagen (Saharia et al., 2022). Similarly, generative models are the core of modern speech synthesis systems (e.g. Kim et al. (2020); Popov et al. (2021); Tinchev et al. (2023)) and music generation (Oord et al., 2016; Yang et al., 2017). Apart from their creative usage, the same methods are also used for scientific purposes. Several works employ recent generative models to speed up the process of physical processes simulations (Deja et al., 2018; Paganini et al., 2018) or the discovery of new drugs and molecules (Gupta et al., 2018; Blaschke et al., 2018).

The above-mentioned practical applications of generative models are usually based on one of their fundamental properties – the possibility of sampling new data points from the approximated training data distribution. While this is a significant feature of generative models, in this work, we focus on the fact that in the process of their training, generative models identify patterns hidden in data and learn meaningful data representations that might be interesting to analyse and valuable for different downstream tasks. In particular, latent variable models (e.g. VAE) directly encode data samples such as images into lower-dimensional vectors known as *hidden factors* that are later used for sampling and decoding into input data space. Normalising flows (Rezende and Mohamed, 2015) including Glows (Kingma and Dhariwal, 2018) are explicitly trained to map original data samples into a data manifold through invertible operations. Even GANs designed without any notion of latent representation are known to implicitly learn data representations by organising the input noise space in a meaningful way (Zhu et al., 2016; Creswell and Bharath, 2018). Finally, recently proposed diffusion models generate new samples in small steps, gradually removing random noise through numerous inner states with temporal representations.

A common concept of creating a meaningful, usually low-dimensional space that

encodes individual features describing the original input data exists in all those methods. For example, in terms of image, such features can describe the general representation of human hair (its colour, length or style) instead of the individual pixel values composing the whole haircut. Building such representations allows for structuring knowledge devised from data that can be applied in downstream tasks, or consolidated over time.

1.1. Research Questions

In this work, we focus on the inner-workings of different generative models tackling three main research questions:

1. *How can we encode data examples into representations useful for generative modelling?*
2. *Can the internal data representations learned in an unsupervised way by generative models be useful outside of the generative modelling task?*
3. *How do the internal data representations change when retraining the model with additional data? Can we consolidate the knowledge coming to the generative model in separate tasks by aligning the latent data representations?*

In the following chapters, we tackle the enlisted research questions, grouping them into three parts as presented in Figure. 1.1.1. In the first group, called *structure*, we focus on the problem of how to encode data examples into representations so that they are useful in generative tasks. The most common approach to this problem, employed, for example, in VAE, is to encode data features into prior (e.g. Gaussian) distribution. We overview such methods in Chapter 6.3. At the same time, in this thesis, we propose and analyse several alternatives such as learnable prior (Chapter 4), a diffusion process of learning representations (Chapters 5 and 6) and binary latent representations (Chapter 8).

In the second group referred to as *applications*, we discuss possible use cases where data representations trained in an unsupervised way might be useful outside of the generative modelling task. Numerous works described in Sec. 3.1.5 employ this approach for tasks such as clustering, querying, and visualisation. As a part of this thesis, we present our solutions where we use latent representations from generative models in classification, semi-supervised learning, domain adaptation (Chapter 6) and data compression (Chapter 8).

Finally, in the last group called *consolidation*, we move to the continual learning setup. In this machine-learning paradigm, a model incrementally learns from a stream of data over time without losing the ability to perform well on previously

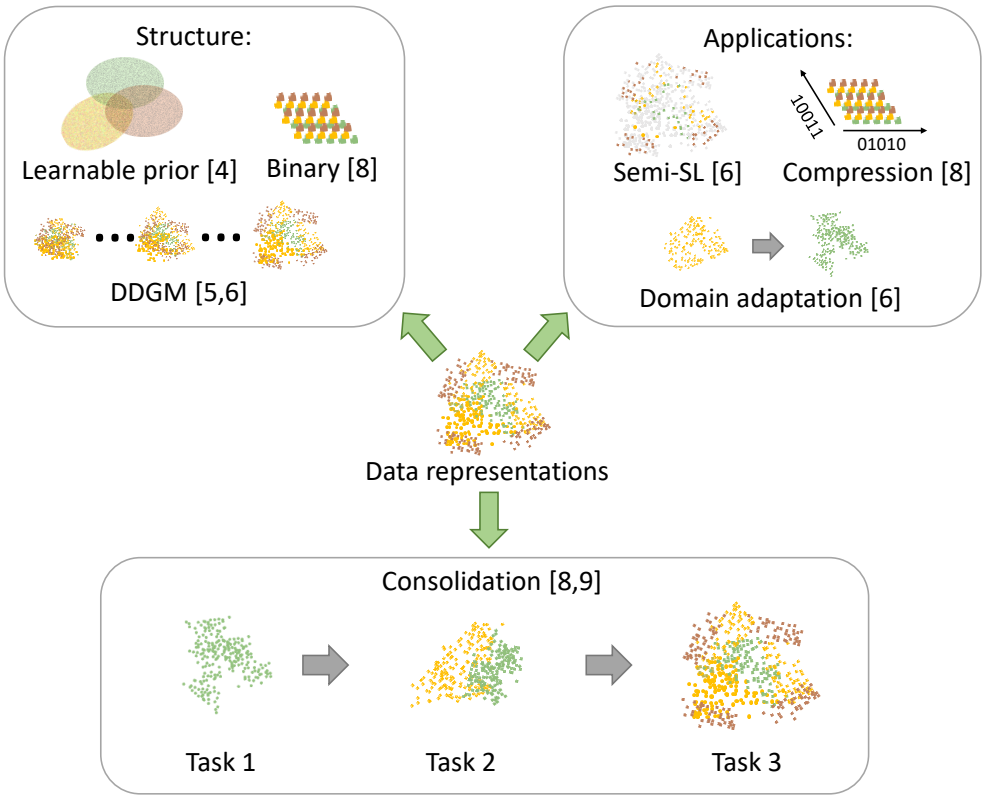


Figure 1.1.1. In this thesis we focus on the data representations in generative models. In the series of five works, we overview their structure, possible applications and usability in continuous knowledge consolidation. Here, we show a graphical overview of this work, where individual publications (indicated as Chapters in the brackets) are grouped in those three main categories.

seen tasks. In Chapter 7, we formally introduce this setup and overview the works on this topic. Once more, we focus on the generative models and their latent data representations. In Chapters 8 and 9, we present our two methods that consolidate knowledge coming to the model in parts, using the latent representations of the generative models.

1.2. Thesis Contribution

The remaining of this work is structured as follows. In the next two chapters, we introduce the background and related works on generative modelling. We pay special attention to the representations within the existing methods. Similarly, in Chapter 7, we introduce the problem of continual learning and the corresponding existing methods on this topic. In Chapters 4, 5, 6, 8 and 9, we present the **core contributions of this thesis** presented as a series of five publications described below:

1.2.1. Generative autoencoder with limited latent space regularisation

Publication: Deja Kamil, Jan Dubiński, Piotr Nowak, Sandro Wenzel, Przemysław Spurek, and Tomasz Trzciński. "End-to-end sinkhorn autoencoder with noise generator." IEEE Access 9 (2020).

In the first work (Deja et al., 2020), presented in Chapter 4, we focus on the structure of data representations in the latent space of a generative autoencoder. We adhere to the problem of prior and posterior mismatch in generative autoencoders, which is a source of low-quality generations. This issue arises when data representations learned by a stochastic encoder are not perfectly regularised to the fixed prior distribution used for sampling of the new examples.To mitigate this issue, we propose a generative autoencoder that learns internal data representations without regularisation. At the same time, we employ an additional neural network to learn the mapping between the random Gaussian noise and the autoencoder's latent space, as presented in Figure 1.2.1. We use the Sinkhorn approximation of the Wasserstein Distance to align a noise mapped through a non-linear multi-layered perceptron with the encoded data representations. Our formulation with learnable prior allows us to better cover the posterior distribution for more complex datasets.

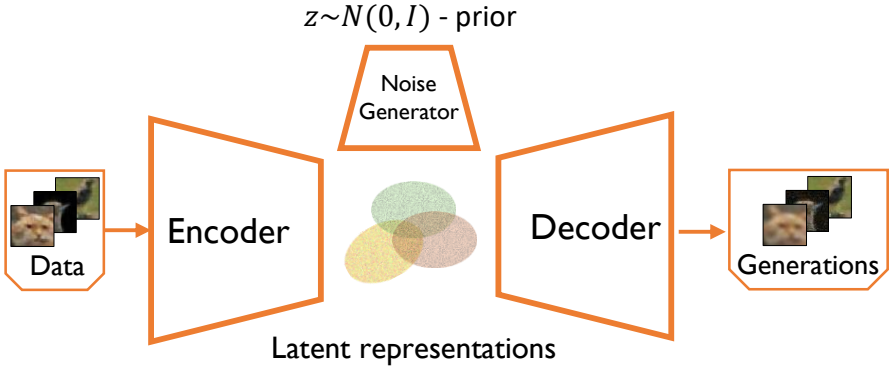


Figure 1.2.1. We propose an End-to-end Sinkhorn autoencoder with noise generator, where data representations are encoded freely into the latent space, while the prior distribution is aligned through the additional MLP trained with a sinhorn approximation of the Wasserstein Distance.

With a series of experiments, we show that our model can achieve state-of-the-art performance compared to other generative autoencoders. We evaluate our method on standard benchmarks such as MNIST or CelebA and extend it to the real-case scenario of High Energy Physics simulations.

The PhD Candidate, as the primary author, devised, implemented, and tested the proposed method. Collaborating with co-authors, including physicists and ex-

perts affiliated with CERN, the PhD Candidate applied the idea in High Energy Physics domain, to efficiently simulate the response of detectors to particle collisions at CERN.

1.2.2. Analysis of the generative process in DDGMs

Publication: Deja Kamil, Anna Kuzina, Tomasz Trzciński, and Jakub M. Tomczak. "On analyzing generative and denoising capabilities of diffusion-based deep generative models." Advances in Neural Information Processing Systems 35, 2022, (NeurIPS22).

In the second work (Deja et al., 2022a), presented in Chapter 5, we move to the recently proposed Deep Generative Diffusion Models (DDGM). We once more focus on the structure of data representations in generative models, this time understood as temporal images refined by the backward diffusion process of DDGMs. We start with an extensive analysis of this family of methods and evaluate how data points generated by the backward diffusion process change with time. Our experiments show an interesting insight that the decoder model of the DDGM changes its behaviour depending on the diffusion timesteps. We observe that at the early stage of the backward diffusion process (closer to the noise), the model creates new data features similar to those observed in the original training data. Simultaneously, the same model applied to the late steps (closer to the final image) changes into a standard denoising autoencoder and removes the remaining noise artefacts in a data-agnostic way. Therefore, we conclude that Deep Generative Diffusion Models can be seen as a combination of two parts – a *generator* that creates new data features from the learned data distribution and a data-agnostic *denoiser*. We visualise this observation in Figure 1.2.2.

Consequently, on top of this observation, we introduce a hybrid model dubbed DAED that consists of a DDGM as a generative part and a simple denoising autoencoder that combines the last steps of the diffusion process. We postulate to model those two networks with separate parameterisations. Through experiments, we validate our proposition and show how it can improve the efficiency of the DDGMs.

The PhD Candidate, as the first author, played a key role in developing and implementing the method, and in developing and conducting the analysis. Working together with co- -authors, the PhD Candidate prepared the manuscript for publication and presented the findings at a conference. It is important to note that this article was prepared during the PhD Candidate's internship at Vrije Universiteit Amsterdam.

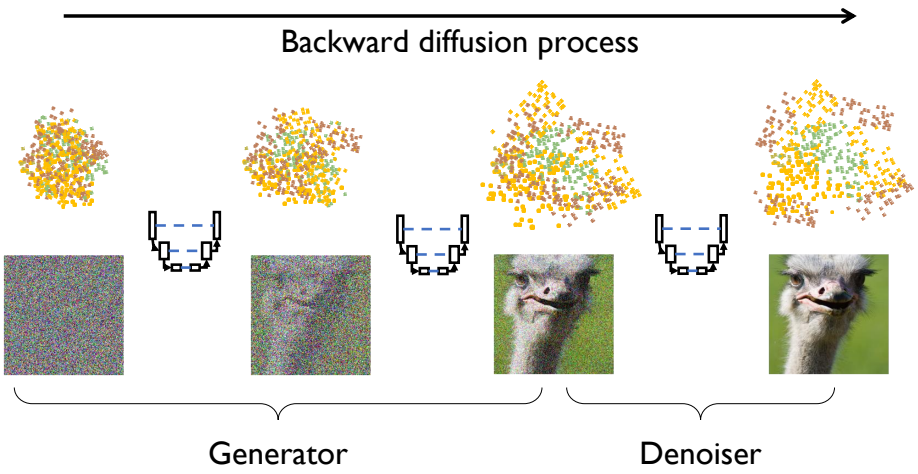


Figure 1.2.2. We analyse how internal states of the diffusion generative models change in the process of image generation. We observe that those methods can be split in two parts – a generator that creates new data features, and data agnostic denoiser.

1.2.3. Data representation in DDGMs

Publication: Deja Kamil, Tomasz Trzciński, and Jakub M. Tomczak. "Learning data representations with joint diffusion models", *accepted for* European Conference on Machine Learning 2023 (ECML 2023).

In the third work (Deja et al., 2023), presented in Chapter 6, we delve further into the previous chapter's analysis of the representations *structure* in DDGMs and extend it also to their *applications*. However, we deepen the analysis, by looking for robust data representations inside the decoder model. Since most works in the topic of diffusion-based generative modelling employ the UNet (Ronneberger et al., 2015) architecture based on the encoder-decoder structure, we propose to treat the output of the encoder part as internal data representations. We visualise this approach in Figure 1.2.3. In the first experiment, we show that such features encoded with a DDGM trained in a fully unsupervised way, contain meaningful information that can be used in downstream tasks such as classification. Moreover, in reference to our previous work described above, we show that those representations change with diffusion timesteps. We can observe that in the early backward diffusion steps (closer to the noise), the model learns low-grain features such as hair colour in the portrait photographs, while the same model applied to the latest steps encode high-fidelity image attributes (e.g. necklace).

We propose to use this observation in a joint diffusion model that incorporates the extracted representations as input for a classifier and optimises the entire network jointly. Therefore, our one model employs a single parameterisation to cap-

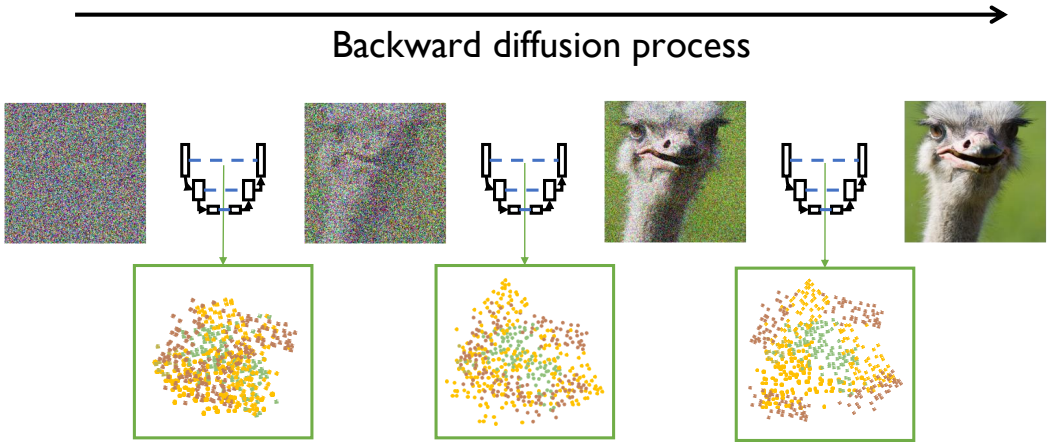


Figure 1.2.3. We propose to investigate latent representations of the denoiser trained as a diffusion-based generative model. We show that those features encode useful information, and propose a joint diffusion model with classifier that can be used for multiple downstream tasks.

ture both the probability of an example and the marginal probability of its class assignment. This improves the classification's performance and allows for generating higher-quality guided samples. In a series of experiments, we also show how we can benefit from such formulation in various applications. In particular, we demonstrate how our joint model can be used in a semi-supervised setup, where learning how to generate unlabelled examples can enhance the quality of shared representations and hence the performance of a classifier trained with limited amount of labelled data. We use a similar approach for domain adaptation task, where we adapt to a new domain with the generative part of our model without supervision. Finally, we show how we can use the generative part of our model to explain the classifier's decision by generating counterfactual explanations.

This work was primarily devised by the PhD Candidate who conducted all of the experiments with guidance from two supervisors.

1.2.4. Binary data representations for image compression with autoencoder

Publication: Deja Kamil, Paweł Wawrzyński, Wojciech Masarczyk, Daniel Marczak, and Tomasz Trzciński. "Bin-play: A binary latent autoencoder for generative replay continual learning". 2021 International Joint Conference on Neural Networks (IJCNN21).

In the fourth work (Deja et al., 2021), presented in Chapter 8, we evaluate alternative – binary form of data representations *structure*, its *application* in data

compression and capability of *consolidating* knowledge. In particular, we focus on the role of generative models in continual learning as an efficient method for storing past data examples useful for rehearsal. We present our method called BinPlay, where we explore the possibility of using latent representations as the efficient data compression mechanism. We first observe that current methods based on standard methods such as VAE or GANs tend to lose the quality of the stored examples when trying to generalise outside of the training set. Therefore, in our work, we leverage a binary autoencoder for storing data examples efficiently as presented in Figure 1.2.4. The goal of the autoencoder is to encode past samples into a set of predefined binary codes living in its binary latent space. By parameterising the code computation formula solely on the chronological indices of the training samples, our system can compute the binary codes of rehearsed samples on-the-fly, without a need to store them in the memory.

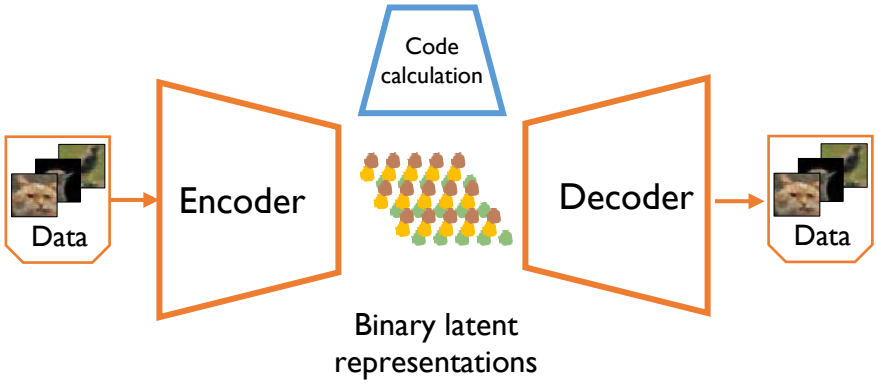


Figure 1.2.4. We introduce BinPlay – a binary autoencoder for efficient storage of past data examples in the problem of continual learning.

Our experiments show that such an approach provides a very efficient way for storing past data examples without a drop in their quality usually observed in other generative rehearsal methods. Thanks to that, we can observe a significant improvement in the continually trained system when rehearsed with high-quality samples.

As the primary author of this publication, the PhD Candidate took the lead in implementing and evaluating the method devised together with the co-authors. The collaborative efforts resulted in the the manuscript that the PhD Candidate had the privilege of presenting at a conference.

1.2.5. Continuous knowledge consolidation in variational autoencoder's latent space

Publication: Deja Kamil, Paweł Wawrzyński, Wojciech Masarczyk, Daniel Marczak, and Tomasz Trzciński. "Multiband VAE: Latent Space Alignment for Knowl-

edge Consolidation in Continual Learning.", Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI22).

Finally, in Chapter 9, we present the fifth work (Deja et al., 2022b) which focuses directly on the knowledge *consolidation* in the latent representations of generative model. To that end, we tackle the problem of continual learning of the variational autoencoder. We first observe that existing methods evaluated primarily in artificial scenarios fail to consider real-life situations where subsequently presented data portions might share some similarity. Therefore, we propose to posit the goal of continual learning of the generative model as a knowledge accumulation task. To formalise this setup, we introduce a new knowledge consolidation scenario where consecutive tasks share partially similarity by following the Dirichlet distribution of classes. We show that our benchmark is considerably more challenging for the majority of the related methods.

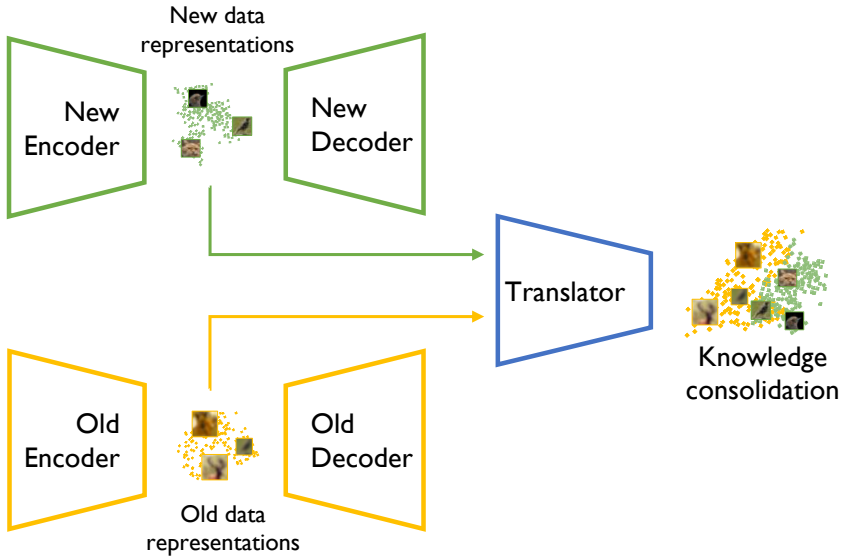


Figure 1.2.5. We propose Multiband VAE - a method for continuous knowledge consolidation through alignment of latent representations in Variational Autoencoder.

To overcome the limitations of recent works, we introduce a Multiband VAE (Deja et al., 2022b) - a method for continual knowledge consolidation through the alignment of the latent representations in the Variational Autoencoder. We postulate that to consolidate knowledge coming to the model in sequence, we should divide the model update process into two parts. In the first one, we learn new data representations, while in the second, we use an additional neural network to map the new data encoding with the previously learned ones as presented in Figure 1.2.5. We also develop a controlled forgetting mechanism to improve the model's plasticity,

allowing for selected samples to be overwritten by similar new data based on their similarity in the latent space.

In the experiments, we show that Multiband VAE significantly outperforms recent state-of-the-art methods in both: standard and the proposed knowledge consolidation scenario. Moreover, we show that our approach can at the same time prevent forgetting past data examples and improve when retrained with partially similar data. To our knowledge this is the first work where we can observe such behaviour in continual generative learning.

The PhD Candidate, as the first author of this publication, played a crucial role in the development, implementation, and evaluation of the proposed method. Specifically, the PhD Candidate took responsibility for implementing the algorithm, conducting ablation studies, and comparing the results to related works. Working collaboratively with co-authors and supervisors, the PhD Candidate prepared the manuscript for publication and had the opportunity to present it in-person at a conference.

1.3. Publications Not Included in the Dissertation

1. **Kamil Deja**, Georgi Tinchev, Marta Czarnowska, Marius Cotescu, Jasha Droppo. "Diffusion-based accent modelling in speech synthesis.", *accepted for* Interspeech 2023, (2023).
2. Zając Michał, **Kamil Deja**, Anna Kuzina, Jakub M. Tomczak, Tomasz Trzciński, Florian Shkurti, and Piotr Miłoś. "Exploring Continual Learning of Diffusion Models.", Continual Learning Vision CVPR Workshop, (2023).
3. Tinchev Georgi, Marta Czarnowska, **Kamil Deja**, Kayoko Yanagisawa and Marius Cotescu, "Modelling low-resource accents without accent-specific TTS frontend.", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (2023).
4. **Deja Kamil**, Ariadna Sanchez, Julian Roth, Marius Cotescu. "Automatic Evaluation of Speaker Similarity.", Interspeech 2022, (2022)
5. Dubiński Jan, **Kamil Deja**, Sandro Wenzel, Przemysław Rokita, and Tomasz Trzciński. "Selectively increasing the diversity of GAN-generated samples.", 29th International Conference on Neural Information Processing (ICONIP), (2022).
6. Masarczyk Wojciech, Paweł Wawrzyński, Daniel Marczak, **Kamil Deja** and Tomasz Trzciński. "Logarithmic continual learning.", IEEE Access 10, (2022).
7. Graczykowski Łukasz, Monika Jakubowska, **Kamil Deja**, Maja Kabus, and the ALICE Collaboration. "Using machine learning for particle identification in ALICE." Journal of Instrumentation, (2022).
8. Masarczyk Wojciech, **Kamil Deja**, and Tomasz Trzciński . "On robustness of generative representations against catastrophic forgetting.", 28th International Conference on Neural Information Processing (ICONIP), (2021).
9. **Deja Kamil**, Tomasz Trzciński, Łukasz Graczykowski, and the ALICE Collaboration. "Generative models for fast cluster simulations in the TPC for the ALICE experiment.", Information Technology, Systems Research, and Computational Physics, (2020)
10. **Deja Kamil**. "Using Machine Learning techniques for Data Quality Monitoring in CMS and ALICE experiments.", Large Hadron Collider Physics (LHCP), Proceedings of Science, (2019).
11. Trzciński Tomasz, and **Kamil Deja**. "Assigning quality labels in the high-energy physics experiment ALICE using machine learning algorithms.", Acta Phys. Polon. Suppl., (2018).

Additional 130 co-authored articles published as a member of the ALICE Collaboration.

2. Background

2.1. Generative Autoencoders

In this work, we propose to look at the generative autoencoders as a particular case of the standard autoencoders- a type of neural network used for unsupervised learning of data representations. The vanilla autoencoder consists of two parts: an encoder q_ϕ that maps input data into the lower-dimensional representation $\mathbf{z}$, and a decoder p_θ that decodes the representation back into original data space. The encoder and decoder are typically trained to minimise the reconstruction error between the input and the output

$$L_r = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2, \tag{2.1}$$

where $\hat{\mathbf{x}} = p_\theta(q_\phi(\mathbf{x}))$. Due to the bottleneck in the autoencoder's architecture where latent representation $\mathbf{z}$ is of lower dimensionality than the input, autoencoders learn to group semantically meaningful data representations that might be used for several tasks such as dimensionality reduction (Wang et al., 2016), representation learning (Vincent et al., 2010), image compression (Theis et al., 2017), anomaly detection (Sakurada and Yairi, 2014; Alain and Bengio, 2014; Zhou and Paffenroth, 2017; Pol et al., 2019; Deja, 2019) or image search (Strub et al., 2016). Despite their usefulness in different domains, vanilla autoencoder models trained with sole reconstruction loss cannot be directly used as generative models. First of all, deterministic autoencoders can not describe continuous data distribution. Secondly, without any regularisation, vanilla encoders encode examples into non-overlapping distributions with discontinuities. While such formulation enables accurate reconstructions, sampling from the created latent space is nearly impossible.

2.1.1. Variational Autoencoder

Therefore, in the Variational Autoencoder (VAE), Kingma and Welling (2014) introduce a probabilistic autoencoder that enforces the multidimensional Gaussian distribution in the autoencoder's latent space. To that end, the output of the encoder network is split into two parts. The first defines the mean of the Gaussian distribu-

tion μ, and the second one is its variance σ^2. This changes the model into stochastic encoder $q_\phi(\mathbf{z}|\mathbf{x})$, that maps every single training example into the Gaussian distribution $\mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x})I)$. Additionally, the authors propose to change the deterministic decoder into a stochastic version that models the marginal data distribution with respect to the sampled latent representations $p(\mathbf{x}|\mathbf{z})$. With such formulation, the reconstruction error of the Variational Autoencoder can be understood in a probabilistic way as a maximization of the likelihood of $\mathbf{x}$ given a latent variable $\mathbf{z}$:

$$L_r = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[lnp(\mathbf{x}|\mathbf{z})] \tag{2.2}$$

Changing the autoencoder model into a probabilistic one resolves the problem of gaps in the standard autoencoder's latent space since every training example can result in multiple latent representations $\mathbf{z}$ – and therefore occupy the area of latent space instead of a single point. The remaining problem is to ensure that all of the representations are arranged in the well-defined part of the hyperplane, so that we can easily sample new instances. To that end, the authors propose a regularisation in the form of Kullback-Leibler divergence that regularises the latent Gaussian distribution to the Normal one. Therefore, the new training objective of the VAE is a combination of reconstruction error and the regularisation term:

$$L = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[lnp(\mathbf{x}|\mathbf{z})] - KL[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})], \tag{2.3}$$

where $p(\mathbf{z})$ is a pre-defined *prior* distribution - usually a Normal Gaussian.

Finally, as a part of changing the vanilla autoencoder into the probabilistic model, authors propose to sample latent representation $\mathbf{z}$ from variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$. Unfortunately, this procedure breaks the gradient flow through the neural network. To circumvent this issue in VAE authors use a *reparametrisation trick* where a source of randomness is extracted outside the neural network. In particular, instead of sampling from the predicted Gaussian distribution $\mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x})I)$, authors propose first to sample a random noise $\epsilon \sim \mathcal{N}(0, I)$ and then calculate a sampled latent representation $\mathbf{z} = \mu(\mathbf{x}) + \epsilon \cdot \sigma(\mathbf{x})$. By moving the sampling outside of the VAE flow, this trick enables the model's training with error backpropagation.

2.2. Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014a) implement the conceptually opposite approach to generative modelling. Instead of encoding examples into latent space from where they can be decoded or generated, GANs are trained to directly sample from the prior distribution $\mathbf{z} \sim \mathcal{N}(0, I)$ and then gen-

erate new instances. To that end, a model called *generator* G_θ is used to turn the random noise into an object $\mathbf{x}$. Since the generator creates new examples from randomly sampled noise, it is impossible to train the model with reconstruction error. Instead Goodfellow et al. (2014a) introduce an adversarial training with an additional neural network called *discriminator* D_ϕ. The generator and discriminator are trained simultaneously in a minimax game, where the first one tries to generate samples that are indistinguishable from the real ones, while the second one tries to distinguish between real and generated samples correctly. Precisely, we can formalise that the discriminator solves the classification task by assigning 0 to all fake data points and 1 to the real ones. Therefore, the loss of a discriminator can is a binary cross entropy in the following form:

$$L = \mathbb{E}_{\mathbf{x}\sim p_{\text{real}}}\left[\log D_\phi(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(1 - D_\phi(G_\theta(\mathbf{z}))\right)\right], \tag{2.4}$$

where the left part is related to the performance on real data examples, while the right the quality of generator in distinguishing the fake ones. Since the generator tries to create images that are as similar to the original data as possible, its training objective is to maximise the training objective of the discriminator. Therefore we can write the following learning objective of the whole system:

$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x}\sim p_{\text{real}}}\left[\log D_\phi(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(1 - D_\phi(G_\theta(\mathbf{z}))\right)\right] \tag{2.5}$$

Because of the minimax game, the parameters of the generator and discriminator have to be optimised in turns. In such a case, the decoder better distinguishes fake and real examples. At the same time, the generator trained with a gradient from constantly improving discriminator also progresses in generating examples that mimic the original data samples with increasing accuracy.

Adversarial training has several advantages over the reconstruction error. Most importantly, it does not suffer from the problem of averaged reconstructions, and therefore GANs usually produce sharper and more exact images than VAEs. Nevertheless, there are two main drawbacks of this approach. Firstly, the minimax-based training is unstable and often results in suboptimal solutions known as mode collapse (Bau et al., 2019). Secondly, the GAN model does not have the capability of encoding original data examples into meaningful representations. Therefore, this work rarely refers to this family of methods.

2.3. Diffusion-Based Deep Generative Models (DDGMs)

The last group of generative models we consider in this work is Diffusion-based Deep Generative Models (DDGMs). In this thesis, we follow their formulation as presented in (Ho et al., 2020; Sohl-Dickstein et al., 2015). DDGMs could be seen as infinitely deep hierarchical VAEs with a specific family of variational posteriors (Huang et al., 2021; Kingma et al., 2021; Tomczak, 2022; Tzen and Raginsky, 2019), namely, Gaussian diffusion processes (Sohl-Dickstein et al., 2015). Given a data point $\mathbf{x}_0$ and latent variables $\mathbf{x}_t, \ldots, \mathbf{x}_T$, we want to optimize the marginal likelihood $p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_0, \ldots, \mathbf{x}_T) \mathrm{d}\mathbf{x}_1, \ldots, \mathbf{x}_T$. We define the *backward* (or *reverse*) *process* as a Markov chain with Gaussian transitions starting with $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$, that is:

$$p_\theta(\mathbf{x}_0, \ldots, \mathbf{x}_T) = p(\mathbf{x}_T) \prod_{t=0}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \tag{2.6}$$

where $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$. Additionally, we define the *forward diffusion process* as a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, ..., \beta_T$, namely, $q(\mathbf{x}_1, \ldots, \mathbf{x}_T|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$, where $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$. Let us further define $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{i=0}^{t} \alpha_i$. Since the conditionals in the forward diffusion can be seen as Gaussian linear models, we can analytically calculate the following distributions:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1-\overline{\alpha}_t)\mathbf{I}), \tag{2.7}$$

and

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}), \tag{2.8}$$

where $\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\mathbf{x}_t$, and $\tilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$. We can use (2.7) and (2.8) to define the variational lower bound as follows:

$$\begin{aligned} \ln p_\theta(\mathbf{x}_0) \geq L_{vlb}(\theta) := & \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\ln p_\theta(\mathbf{x}_0|\mathbf{x}_1)]}_{-L_0} - \underbrace{D_{\mathrm{KL}}[q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T)]}_{L_T} \\ & - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} D_{\mathrm{KL}}[q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)]}_{L_{t-1}}. \end{aligned} \tag{2.9}$$

that we further optimize with respect to the parameters of the backward diffusion.

The conditional likelihood In this paper, we focus on images, thus, data is represented by integers from 0 to 255. Following Ho et al. (2020), we scale them linearly to $[-1,1]$. As a result, to obtain discrete log-likelihoods, we consider the discretized (binned) Gaussian conditional likelihood (Ho et al., 2020):

$$p_\theta(\mathbf{x}_0|\mathbf{x}_1) = \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}\left(x; \mu_\theta^i(\mathbf{x}_1, 1), \sigma_1^2\right) \mathrm{d}x, \tag{2.10}$$

where D is the data dimensionality of $\mathbf{x}_0$, and i denotes one coordinate of $\mathbf{x}_0$, and:

$$\delta_+(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases} \quad \delta_-(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases}. \tag{2.11}$$

Noise scheduling Originally, Ho et al. (2020) propose to linearly scale the noise parameters β_t (*linear scheduling*), e.g., scaling linearly from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. In Nichol and Dhariwal (2021), authors suggest to increase the number of less noisy steps through *cosine scheduling*: $\bar{\alpha}_t = \frac{f(t)}{f(0)}$, $f(t) = \cos\left(\frac{t/T+c}{1+c} \cdot \frac{\pi}{2}\right)^2$, $c > 0$ with clipping the values of β_t to 0.999 to prevent potential instabilities at the end of the diffusion.

Training details In Ho et al. (2020), authors notice that a single part of the variational lower bound is equal to:

$$L_t(\theta) = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \overline{\alpha}_t)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\overline{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \overline{\alpha}_t} \boldsymbol{\epsilon}, t \right) \right\|^2 \right], \tag{2.12}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\boldsymbol{\epsilon}_\theta$ is a neural network predicting the noise $\boldsymbol{\epsilon}$ from $\mathbf{x}_t$. Since we use (2.8) in the variational lower bound objective (2.9), and $\mathbf{x}_t$ could be sampled from the forward diffusion for a given data, see (2.7), we can optimise one layer at a time. In other words, we can randomly pick a specific component of the objective, L_t, and update the parameters by optimising L_t without running the whole forward process from $\mathbf{x}_0$ to $\mathbf{x}_T$. As a result, the training becomes very efficient and learning very deep models (with hundreds or even thousands of steps) is possible.

In Ho et al. (2020), it is also proposed to train a simplified objective that is a version of (2.12) without scaling, namely:

$$L_{t,\text{simple}}(\theta) = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\overline{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \overline{\alpha}_t} \boldsymbol{\epsilon}, t \right) \right\|^2 \right], \tag{2.13}$$

where t is uniformly sampled between 1 and T. To further reduce computational and memory costs, typically, a single, shared neural network is used for modelling $\boldsymbol{\epsilon}_\theta$ Ho

et al. (2020); Kingma et al. (2021); Nichol and Dhariwal (2021) that is parameterised by an architecture based on U-Net type neural net Ronneberger et al. (2015). The U-Net could be seen as a specific auto-encoder that passes all codes from the encoder to the decoder.

3. Related Works

In this chapter, we overview works related to our main contributions. We first focus on generative autoencoders and then move to the overview of novel methods based on Diffusion-based generative models.

3.1. Generative Autoencoders

In the previous chapter, we introduced the general concept of generative autoencoder using a Variational Autoencoder as an example. In this section, we first overview other common approaches implementing the same idea with some modifications. We then present selected methods that focus on latent representations from generative autoencoders and use them for different tasks.

3.1.1. Latent space regularisation in generative autoencoders

In Chapter. 3.1, we introduced Variational Autoencoder as the most popular implementation of generative autoencoder. In VAE, an encoder maps data samples into the latent space and decodes them while regularising the latent representation to follow Normal distribution using a KL-divergence. Several works implement the same principle differently, as overviewed by Chadebec et al. (2022). We summarise those different approaches in Table 3.1.1 and briefly describe them in this section.

Following the works on Wasserstein GAN (Arjovsky et al., 2017), Tolstikhin et al. (2017) introduces a Wasserstein Autoencoder (WAE), where the regularisation function is changed to Wasserstein distance. In particular, Tolstikhin et al. (2017) introduce two possible methods for applying Wasserstein distance on the autoencoder's latent space. The first one is based on the *maximum mean discrepancy* (Gretton et al., 2012) (MMD) technique, while the second one, similarly to Arjovsky et al. (2017), uses a neural network as a critic. In the latter case, the encoder is trained to store data examples in the latent space as close as possible to the prior distribution with respect to the decision from the adversarially trained external module.

Similarly, Kolouri et al. (2018) present the Sliced-Wasserstein Autoencoder (SWAE), which substitutes MMD with an approximation obtained by a cumulative distribution of one-dimensional distances. The main innovation of SWAE is

the sliced-Wasserstein distance, a fast-to-estimate metric for comparing two distributions based on the mean Wasserstein distance of one-dimensional projections. This solution is much simpler, but as reported by Patrini et al. (2019), it results in a lower diversity of generated results. Knop et al. (2020), present a modification of this method known as the Cramer-Wold AutoEncoder (CWAE), where the sliced Wasserstein distance is replaced with the CW-distance between distributions. CWAE model can be seen as a version of the WAE-MMD method with a choice of a specific Cramer-Wold kernel.

One more approximation of Wasserstein Distance is introduced by Patrini et al. (2019) in the Sinkhorn Autoencoder (SAE). This work's approximation is based on the p-Wasserstein distance calculated in a latent space via backpropagation through the Sinkhorn algorithm (Cuturi, 2013). Thus, SAE can work with different metric spaces and priors with minimal adaptations. In particular, Patrini et al. (2019) experiment with the normal and hypersphere prior.

Finally, in Adversarial Autoencoders, Makhzani et al. (2015) apply adversarial training directly in the latent space of the generative autoencoder to enforce alignment between the prior and posterior distribution. In this work, an additional neural network is trained in a min-max game to align encoded examples with prior data distribution.

3.1.2. Generative autoencoders with adversarial training

Apart from the adversarial autoencoders described in the previous section, where adversarial loss was used as the regularisation term for generative autoencoders latent space, there exits a group of method further influenced by the adversarial training, using it instead of the reconstruction loss. Arora et al. (2017) and Arora and Zhang (2017) noticed that training of a standard GAN is unstable and may result in limited quantitative properties, while Variational Autoencoders converge much better but tend to generate blurry samples when applied to natural images. Therefore, several methods try to combine those two approaches, drawing from the best parts of both worlds.

To that end, Zhu et al. (2022) propose Latently Invertible Autoencoder (LIA) architecture, which employs an additional network in the latent space of VAE. The decoder of LIA is first trained as a standard GAN with the invertible network. Then the partial encoder is learned from a disentangled autoencoder by detaching the invertible network from LIA. On the other hand, Pidhorskyi et al. (2020) introduce an autoencoder architecture by modifying the original GAN paradigm. The generator and discriminator are decomposed into two networks. In consequence, we obtain two additional latent spaces, where we add regularisation terms. The model is op-

Table 3.1.1. Overview of regularisation methods in different generative autoencoders. We compare recent methods with our solution described in Chapter 4.

Method	Latent regularisation	Comments
VAE (Kingma and Welling, 2014)	Kullback-Leibler divergence	Variational scheme – Averaged generations – Limited performance on complex datasets
AAE (Makhzani et al., 2015)	Adversarial	Adversarial training with discriminator: + High quality generations – Training instability
WAE (Tolstikhin et al., 2017)	Wasserstein distance (MMD or adversarial approx.)	Two approximations: MMD or Critic + Fast to calculate (MMD) - Unstable training (critic)
SWAE (Kolouri et al., 2018)	Wasserstein distance (Sliced approximation)	Sliced-Wasserstein distance approximation: • Avg. Cost on one-dimensional projections – Assumes independent encoding
SAE (Patrini et al., 2019)	Wasserstein distance (Sinkhorn approximation)	Sinkhorn approximation: – Slow to compute + High quality generations
CWAE (Knop et al., 2020)	Wasserstein distance (Sliced + CW approximation)	Cramer-Wold kernel: + Faster to calculate
e2e SAE (Deja et al., 2020) Chapter 4	Wasserstein distance (Sinkhorn approximation)	Sinkhorn approximation + Trainable prior

timised by adversarial training. Larsen et al. (2016) introduce a similar approach where the decoder of the generative autoencoder is trained directly as a GAN generator.

3.1.3. Hierarchical Variational Autoencoders

In the previous two sections, we discussed generative autoencoders where a single encoder was used to map original data samples $\mathbf{x}$ into a latent variable $\mathbf{z}$ with a single pass through a neural network. Such formulation has a significant property – the variational inference used, for example, in VAE, enforces independence of all latent variables. Therefore different latent dimensions encode potentially independent data features. This idea, further explored in β-vae by Higgins et al. (2017), sounds appealing, but independent data features learned by latent variable models often entangle numerous attributes recognised by humans.

One possible solution to this problem comes with an observation that many concepts we use to describe the world can be organised hierarchically so that the prop-

erties of one feature influence the possible properties of another. E.g., The general property of the building that describes its size influences the style of the windows we can see. In deep generative modelling, this solution is implemented in hierarchical models, e.g., hierarchical variational autoencoders.

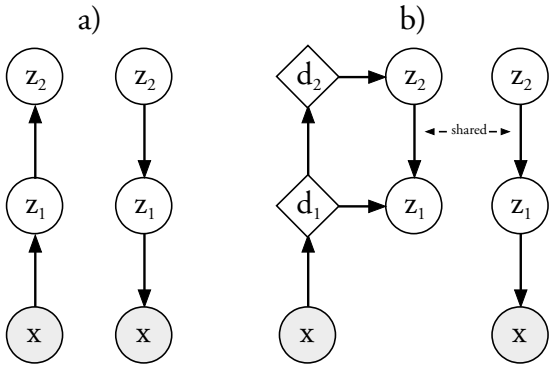


Figure 3.1.1. Encoding and generative models for a) VAE and b) LadderVAE. Circles are stochastic variables and diamonds are deterministic variables. Image from (Maaløe et al., 2017)

In this family of methods, the latent variables are decomposed into multiple levels, where each of them corresponds to a different degree of abstraction or granularity, as presented in Figure. 3.1.1. At each level, the model learns a separate set of latent variables that capture different aspects of the data. The latent variables at each level are assumed to be conditionally independent given the variables at the level above, and the joint distribution over all the variables is modelled using a product of Gaussian distributions. This idea is implemented in the top-down approach proposed in ResNet VAEs (Kingma et al., 2016) and Ladder VAE (Maaløe et al., 2017), and further extended in BIVA (Maaløe et al., 2019), NVAE (Vahdat and Kautz, 2020) and very deep VAE (Child, 2021).

3.1.4. Latent space geometry

There are several publications that focus on the latent space of generative autoencoders. A significant subset of those works relates to the problem of mismatch between the aggregated posterior from the standard VAE and the preset prior. This issue described by Makhzani et al. (2015); Hoffman and Johnson (2016); Tomczak and Welling (2018); Nalisnick et al. (2019a) could be summarised as a situation where there exists a region in the latent space, where aggregated posterior assigns low probability, while at the same time, the prior probability is relatively high. In such cases, generations from the VAE are often of low quality. We run a simple experiment presented in Figure. 3.1.2 to demonstrate this issue. To that end, we train a simple Variational Autoencoder with only two latent variables. We then en-

code all training examples into the latent space to sample the posterior probability distribution. We visualise the encoded mean values of training data examples in Figure 3.1.2a. As we can see, some regions of the latent space have high coverage of training data, while in some areas, there are almost no training examples encoded. Consequently, in Figure 3.1.2b, we present the relatively high-quality generations sampled from high-density regions. Contrary in Figure 3.1.2c, we show generations from the visible *holes* in the latent space that are often blurred and distorted.

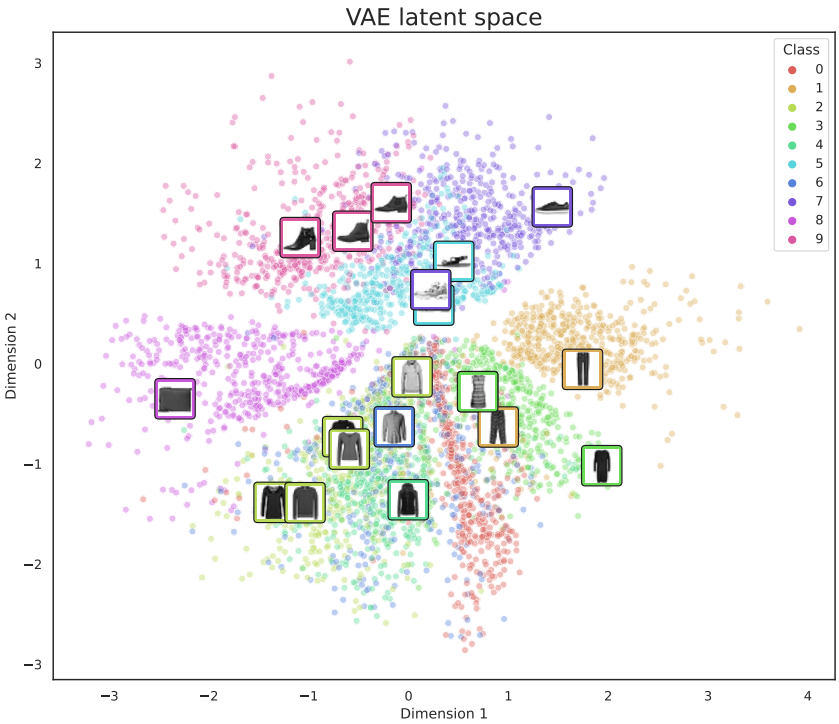
To mitigate this issue, several methods propose learnable priors instead of the fixed Gaussian distribution. The most straightforward implementation of this idea employs a trainable Gaussian Mixture Model (GMM). However, there exist more complex methods. Ziegler and Rush (2019) and Xiao et al. (2019) apply normalising flows to learn the prior distribution in the latent space. With this approach, more complex structures can be approximated through non-linear transformations from flows Gaussian prior. Dai and Wipf (2019) introduce a similar solution with additional VAE in the latent space known as the TwoStageVAE model. As shown by the authors, the second VAE model can correct the mismatch observed in the latent space of the first model. On the other hand, Tomczak and Welling (2018) introduce a new prior that is expressed as a mixture of variational posteriors (VampPrior). The VampPrior consists of a mixture of Gaussians with components conditioned on learnable pseudo-inputs. Such prior is implemented as a two-level hierarchical model.

Recently, a novel method based on DDGMs was proposed to solve the problem related to the prior-posterior mismatch in generative autoencoders. Rombach et al. (2022) introduced a model known as Latent Diffusion, where a standard autoencoder encodes the high-resolution images into the latent space, where a diffusion process runs between prior Gaussian and data representations. The synthesis of new samples with this model consists of two steps: a new representation is first generated through the backward diffusion process, followed by the decoding through the decoder.

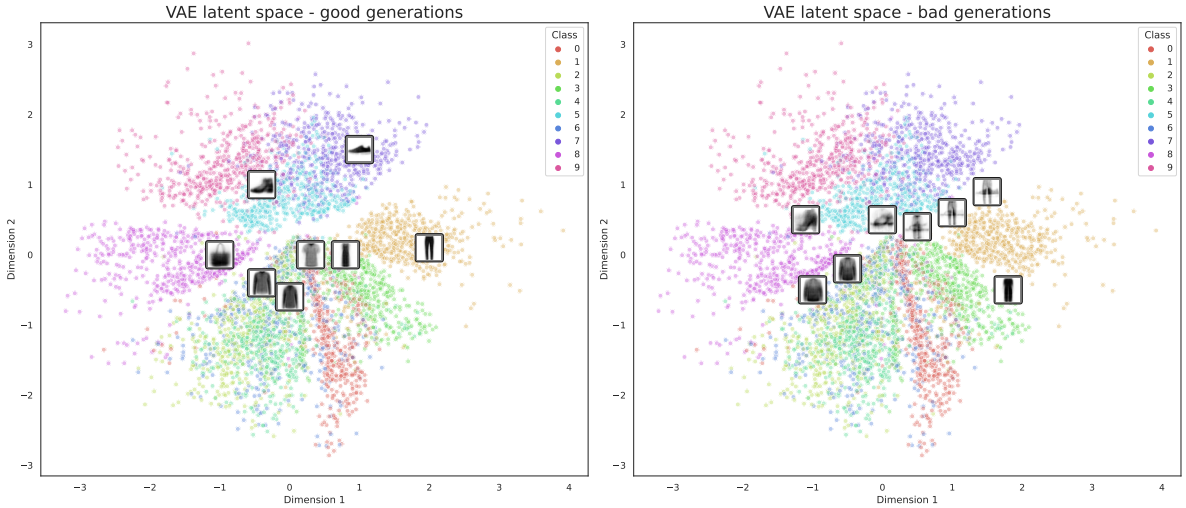
In this thesis, we tackle the problem of prior-posterior mismatch in chapter 4, where we introduce our own method named: end-to-end Sinkhorn Autoencoder. Our approaches alleviate the problem thanks to a learnable prior processed through an additional neural network.

3.1.5. Application of data representations learned with generative models

Apart from the prominent applications where generative models are used for sampling new generations, there are interesting works where the same methods



(a) Encoded training data



(b) Good quality generations from high posterior probability regions

(c) Bad quality generations from low posterior probability regions

Figure 3.1.2. Visualisation of the VAE latent space with two latent variables trained on the FashionMNIST dataset.

are used to extract relevant data features useful for downstream tasks. The most straightforward application of features extracted from generative models is so-called joint or hybrid modelling, where a single model is trained to serve as a generative model and the classifier simultaneously. This means that a single neural network learns data distribution $p(x)$ and the marginal distribution $p(y|x)$. We delve into

this idea in detail in chapter 6, where we introduce our joint diffusion model which uses data representations from a DDGM to perform a classification task.

However, the usage of data features from generative models is also extended to other tasks beyond generative modelling and classification. In particular, Tschannen et al. (2018) overview the role of generative autoencoders in the general area of representation learning. In this domain, an interesting approach is proposed by Burgess et al. (2019) in the MONet model, where a VAE is trained together with a recurrent neural network with an attention mechanism operating in its latent space. This combination is used to decompose 3D scenes into semantically meaningful components. Ding et al. (2021) further explore this technique, employing features from the same architecture for visual reasoning.

Generative models play an essential role in the recent text-to-speech synthesis, while several methods pay special attention to latent data representations features in the generative speech synthesis model. Hsu et al. (2017) introduce a model based on Variational Autoencoder that allows for latent space arithmetic operations. In particular, authors show that by only manipulating the representations, they can change the phonetic content or the speaker identity without parallel training data. Tits et al. (2019) evaluate latent representations from different generative speech synthesis models from the perspective of speech style. In particular, they focus on latent features visualisation and interpretability. From a different perspective, Chung et al. (2016) use sequence-to-sequence autoencoder to learn speech features in an unsupervised way. Qian et al. (2020) and Chan et al. (2022) further explore this idea, using autoencoder to disentangle them into several latent spaces that encode information about language content, timbre, pitch, and rhythm. Such decomposition enables multiple applications such as voice cloning (Wang et al., 2021a), style transfer (Yuan et al., 2021) and mixup (Lee et al., 2021).

There is a growing interest in using generative models for biomedical studies (Wei and Mahmood, 2020), including the specific applications where latent representations are of particular interest. For example, Lopez et al. (2018) propose to use latent representations extracted from a variational autoencoder to cluster information about cells or proteins in single-cell RNA sequencing. This idea was further extended to cancer data integration by Simidjievski et al. (2019). Ding et al. (2018) propose to use variational autoencoders to generate low-dimensionality representations to create interpretable RNA-data representations. The results of this work are presented in Figure 3.1.3. A similar idea is introduced by Szymczak et al. (2022) in HydrAMP, where a conditional VAE is used to learn a lower-dimensional space of peptides' representations in order to capture their antimicrobial conditions.

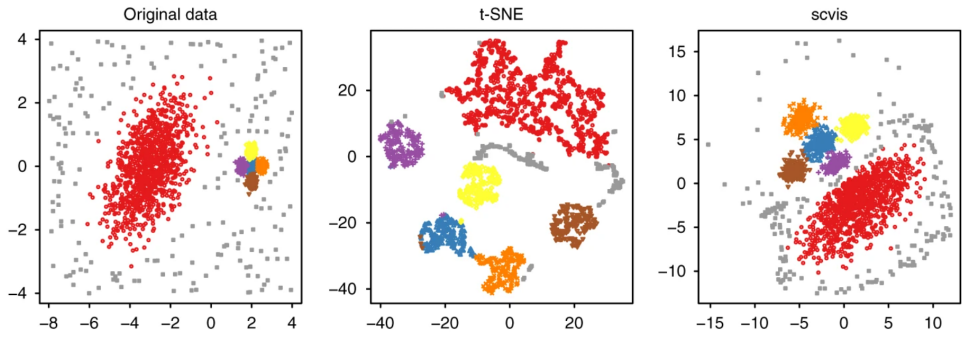


Figure 3.1.3. Comparison of the scvis method with TSNE. The original synthetic data consisted of 2200 points divided into clusters (different colours) with randomly distributed outliers visualised with TSNE (middle) or trained variational autoencoder (right). Image from (Ding et al., 2018).

3.2. Diffusion Based Deep Generative Models

In Chapter 2.3, we introduced the basic setup of DDGMs. However, there are several new extensions that enhance the generative capabilities of DDGMs. For instance, Nichol and Dhariwal (2021) propose technical improvements such as a loss-aware timestep sampling scheduler that selects the timestep for training with a given example based on the accumulated loss value. Additionally, Nichol and Dhariwal (2021) introduce a cosine base noise scheduler, where more noise is added to the input at the latest forward diffusion steps.

Similarly, several works propose to improve the quality of samples from DDGMs by conditioning the generations with class identities (Dhariwal and Nichol, 2021; Tashiro et al., 2021; Ho and Salimans, 2022; Huang et al., 2022). To enhance class-conditioned generations, Dhariwal and Nichol (2021) present a classifier--guided diffusion model. This method incorporates a gradient from an externally trained classifier into the backward diffusion process to guide the generation towards a specific target class. This idea was further extended by Ho and Salimans (2022), where guidance from a classifier was substituted with a two-step synthesis, where features generated unconditionally are removed from the features generated with respect to the target class.

Furthermore, a plethora of applications based on DDGMs, such as DALLE2, Imagen or Midjourney, combine the image generation task with text encoding. The most common approach introduced by Ramesh et al. (2022) is to employ the additional text encoder, usually based on the BERT (Devlin et al., 2019) architecture. The text encoder is aligned with the activations of the DDGM model through the attention mechanism at different UNet levels with a Contrastive Language-Image Pre-training (CLIP) (Radford et al., 2021). Text conditioning greatly improves the

quality of generated samples and allows for prompt-based guidance of the diffusion process.

Song et al. (2020b) present a novel method that approaches diffusion-based modelling from a different perspective. Specifically, the authors develop a diffusion model based on the stochastic differential equation framework, where the diffusion process is continuous and adheres to the Brownian motion principles (Brown, 1828). In this work, instead of directly modelling the density of data distribution, neural network is used to approximate the gradient of the log-likelihood function known as score function. Thanks to the generalisation to SDE framework, in score-based generative models the diffusion process is continuous during training and can be approximated with any number of timesteps during inference.

From yet another perspective, Bansal et al. (2022) introduce a *Cold-Diffusion* method where a diffusion process based on Gaussian noising is substituted with several different deterministic degradation methods such as blur or masking. Surprisingly, authors show that the standard approach generalises well to those techniques, what calls into question the common understanding of DDGMs based on noising.

3.2.1. Connection to hierarchical Variational Autoencoders

Several works have noted the connection of DDGM to VAEs. Huang et al. (2021) focus on the continuous diffusion models and draw the connection to the infinitely deep hierarchical VAEs. Kingma et al. (2021) further explore this path to formulate a VLB objective in terms of the signal-to-noise ratio and propose to learn noise schedule, which brings the forward diffusion process even closer to the encoder of a VAE. In a similar spirit, Vahdat et al. (2021) proposed a latent score-based generative model (LSGM), which can be seen as a VAE with the score-based prior.

3.2.2. DDGMs and data representation

Several works combine Diffusion-Based Generative Modelling with data representation learning. The most pronounced combination is introduced by Rombach et al. (2022), who decomposes the image generation process into two parts as presented in Figure. 3.2.1. First, a standard autoencoder is trained to freely encode and decode the original data into the latent space (without any regularisation). Then, a diffusion process is used to model the generation of image latent representations from prior noise. This work, known as *latent* or *stable* diffusion, achieves comparable results while reducing the computational requirements compared to the standard DDGM. This idea is further extended to a vector quantised diffusion

(VQ-Diffusion Gu et al. (2021)), where a VQ-VAE (van den Oord et al., 2017) instead of a standard autoencoder.

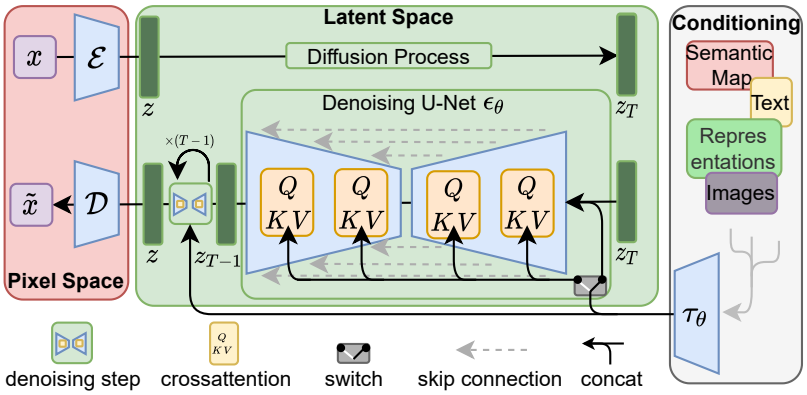


Figure 3.2.1. The overview of the stable diffusion model. An external autoencoder (left) is used to map original images into latent space, where a diffusion process is used to learn data representations. In a process of denoising, a conditioning from text encoder is used through a cross-attention layer at different UNet levels. Image from (Rombach et al., 2022).

There are several works that tackle the problem of learning data representations with DDGMs. Abstreiter et al. (2021) introduce additional encoded information to the score estimator, which allows them to use the score matching loss function for learning data representations. Baranchuk et al. (2021) use activations from the pre-trained diffusion UNet model for the image segmentation task. Other works consider data representations from the UNet model within other generative models. Esser et al. (2018) introduce a conditional UNet-based variational autoencoder, while Falck et al. (2022) show the connection between the UNet architecture and wavelet transformation, applying it to the hierarchical VAEs.

3.3. Generative Models for High Energy Physics

In this work, we focus on the problem of learning representations with generative models. We evaluate our methods in several settings on the most commonly used datasets. Nevertheless, apart from the artificial benchmarks, we also consider a real-world scenario in the High Energy Physics (HEP) domain, which is an interesting test-bed for practical applications of generative modelling. In particular, we consider the problem of simulating detector responses to particle collisions in the ALICE experiment (Aamodt et al., 2008) within the Large Hadron Collider at CERN (Evans and Bryant, 2008a). Recent HEP experiments rely heavily on detailed simulations of the detector responses, as they, compared to the real-world recorded data, allow for accurate reconstruction of the underlying physical phenomena. Traditional approaches to this problem exploit monte-carlo based systems

such as GEANT (Incerti et al., 2018) to calculate every possible interaction between the particle and the experimental apparatus. Although this method provides high-fidelity results, it is at the expense of the extensive computational cost. Therefore multiple attempts are taken to replace the monte-carlo based approach with new – potentially faster techniques such as generative models. The growing interest in this problem resulted in the creation of the Fast Calorimeter Simulation Challenge (Giannelli et al., 2022), where three high-quality datasets with calorimeter responses to particle collisions were prepared. This simplified the data-gathering process and allowed researchers unrelated to physics to contribute to the topic.

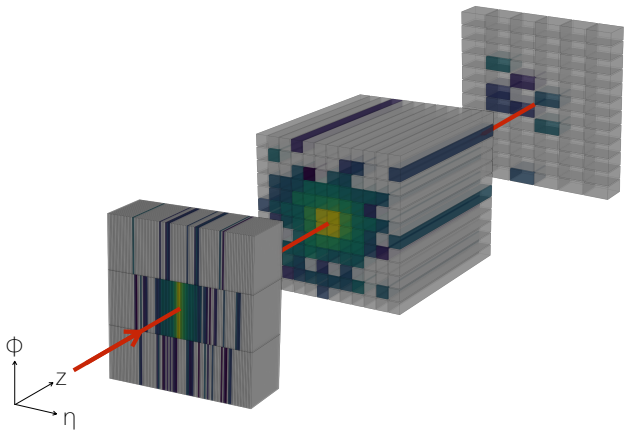


Figure 3.3.1. Visualisation of the calorimeter response to the particle. At different layers of the detector a particle leaves its energy signature visible as voxels of different colours corresponding to the deposited energy. Image from (Paganini et al., 2018).

The majority of current works in this field focus on GAN architectures, e.g. CaloGAN (Paganini et al., 2018) or (Khattak et al., 2018), where a 3D conditional Deep Convolutional GAN architectures are used to simulate the response of the calorimeter of the ATLAS experiment visualised in Figure 3.3.1. This idea is further extended to the simulation of jet pairs by Di Sipio et al. (2019) and Erdmann et al. (2019), where a Wasserstein GAN is used to simulate the calorimeter showers. Similarly, (Kansal et al., 2021) employed GANs with message-passing architecture to generate the cloud of particles. Lately, benefiting from the setup introduced with CaloChallenge, Mikuni and Nachman (2022) introduced a method for fast simulation based on DDGMs with score matching, while Cresswell et al. (2022) proposed a method for calorimeter modelling by first learning their manifold and then estimating the probability density.

In our previous work (Deja et al., 2018), we explore the possibility of simulating particle responses in the TPC detector with different generative models. Our

experiments indicated that by substituting the monte-carlo method with GANs, we could speed up the simulation process by up to two orders of magnitude. In this thesis, we explore the problem of fast simulation of calorimeter response. To that end, in Chapter 4, we introduce a new generative model dubbed end-to-end sinkhorn autoencoder, where we use a sinkhorn algorithm to approximate Wasserstein distance between the encoded examples and noise in the latent space of the generative autoencoder. We propose the method together with its application to the problem of calorimeter response simulation. Similarly to our work, Howard et al. (2022) introduce a fast simulation method based on generative autoencoder, where a Wasserstein distance approximation is used not only to align the latent features of the model with the prior distribution but also to create a mapping between the theoretical models and actual experimental data. Howard et al. (2022) propose using a Sliced Wasserstein Autoencoder to learn the latent space where both simulation and experimental data are encoded simultaneously. This allows for fast simulation of possible detector responses as well as probing and visualisation of the physical properties of encoded real and theoretical data. Moreover, the authors show that their method can be applied in practice by running a real-world physical analysis such as Z-boson and top-quark decays.

3.4. Evaluation of Generative Models

Evaluation of generative models can be a challenging task since they are primarily design to generate data, which makes traditional evaluation methods like accuracy or precision less applicable. Instead, generative models are typically assessed using a combination of quantitative and qualitative metrics.

Since some methods are directly trained to learn the data distribution, one way to evaluate them is by calculating the likelihood of real (usually test) data under the model. Higher likelihood values indicate better performance. However, estimating likelihood can be computationally expensive, especially for complex models such as DDGMs.

Therefore the more common methods involve some forms of perceptual metrics. In particular, Inception Score (IS) by Salimans et al. (2016) employs an externally trained classifier in a form of inception network Szegedy et al. (2015) to extract conditional label distribution $p(y|x)$ for all generated images. Images with meaningful objects should have low entropy on conditional label distribution $p(y|x)$. Moreover, we can measure the variance of generated images as the marginal distribution $\int p(y|x = G(z))dz$.

This methods is further extended to Fréchet Inception Distance (Heusel et al.,

2017), where authors propose to extract features of training and generated data using the Inception model in order to compare them with a Fréchet distance. The closer the extracted features are, the better the generative model's performance.

To disentangle the quality of generated samples from their diversity Sajjadi et al. (2018) introduce precision and recall of the distributions that measures how well the distribution of generated data features fits into the distribution of training examples (Precision), and how well the training data features distribution is covered by the generated data (recall). Those metrics are further improved by Kynkäänniemi et al. (2019).

The FID metric and its extensions are commonly used to evaluate the performance of new methods on well defined benchmarks such as CIFAR10/100, CelebA, or ImageNet. However, as noticed by Kynkäänniemi et al. (2023) the application of those methods to other datasets is limited and biased by the ImageNet classes. Therefore in some cases (e.g. in a speech domain) researchers often relate to human--based perceptual evaluations of generated samples such as Mean Opinion Score (MOS) or Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) tests.

4. End-to-End Sinkhorn Autoencoder With Noise Generator

Title	End-to-End Sinkhorn Autoencoder With Noise Generator
Authors	Kamil Deja, Jan Dubiński, Piotr Nowak, Sandro Wenzel, Przemysław Spurek, and Tomasz Trzciński
Journal	IEEE Access
Volume	9
Year	2020
DOI	10.1109/ACCESS.2020.3048622
Pages	7211 - 7219

Preface

In the previous sections, we overviewed recent works on generative models from the perspective of data representations. In detail, we discussed generative autoencoders that encode original data samples into the latent space while regularising it to follow desired parameterised distribution (e.g. Normal). This enforces the model to encode different data features with independent Gaussian variables. While such an approach is conceptually appealing, it can rarely be successfully implemented in practice.

We describe this problem in section 3.1 and highlight that a prior fixed to the particular distribution might result in *holes* in the latent space of the generative model with low posterior but high prior probability. Consequently, this issue reduces the quality of the generations as the reconstructions sampled from the affected areas are usually imperfect interpolations between significantly different original data examples.

In this chapter, we focus on this problem and the general concept of the data representation structure in the latent space of a generative autoencoder. We evaluate the representations and the methods for their regularisation to the prior distribution. To overcome the problem of prior-posterior mismatch, we introduce an End-to-End Sinkhorn Autoencoder with Noise Generator – a generative autoencoder in which we propose an architecture that separates the process of encoding data samples into latent representations from the process of sampling new instances. In particular, we introduce an additional neural network that maps fixed Gaussian prior to the freely encoded latent representation of the autoencoder. We employ a Sinkhorn approximation of the Wasserstein distance as a regularisation term that aligns encoded data examples with mapped noise. Our experiments show that our approach can alleviate the problem of holes in the generative autoencoder's latent space, which improves the quality of generated samples.

Abstract

In this work, we propose a novel end-to-end Sinkhorn Autoencoder with a noise generator for efficient data collection simulation. Simulating processes that aim at collecting experimental data is crucial for multiple real-life applications, including nuclear medicine, astronomy, and high energy physics. Contemporary methods, such as Monte Carlo algorithms, provide high-fidelity results at a price of high computational cost. Multiple attempts are taken to reduce this burden, e.g. using generative approaches based on Generative Adversarial Networks or Variational Autoencoders. Although such methods are much faster, they are often unstable in training and do not allow sampling from an entire data distribution. To address these shortcomings, we introduce a novel method dubbed *end-to-end Sinkhorn Autoencoder*, that leverages the Sinkhorn algorithm to explicitly align distribution of encoded real data examples and generated noise. More precisely, we extend autoencoder architecture by adding a deterministic neural network trained to map noise from a known distribution onto autoencoder latent space representing data distribution. We optimise the entire model jointly. Our method outperforms competing approaches on a challenging dataset of simulation data from Zero Degree Calorimeters of ALICE experiment in LHC, as well as standard benchmarks, such as MNIST and CelebA.

4.1. Introduction

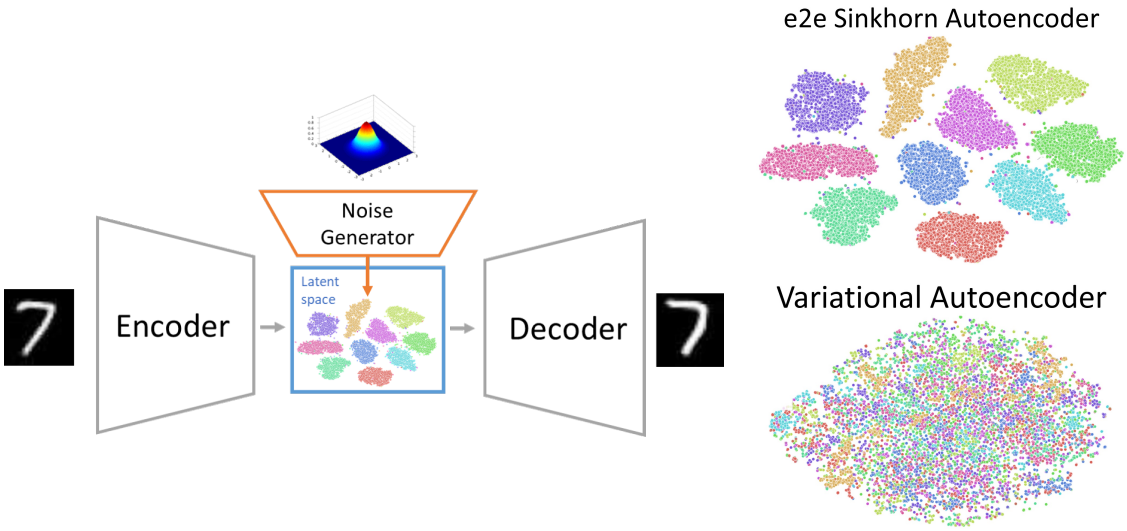


Figure 4.1.1. Schematic visualisation of end-to-end Sinkhorn Autoencoder processing (left). T-SNE visualisation of latent space for MNIST dataset (right). Our conditional e2e Sinkhorn Autoencoder (top) and conditional VAE (bottom). Our model does not restrict latent space to the normal distribution, therefore classes may be even linearly separable.

Multiple real-life applications rely heavily on detailed simulations of ongoing processes, from atomic structures in nuclear medicine (e.g. tomography) (Strulab et al., 2003) or genetics (Incerti et al., 2018), to astrophysics (Zoglauer et al., 2006). This is also true for the Large Hadron Collider (LHC) (Evans and Bryant, 2008b) – one of the biggest scientific programmes currently being carried out worldwide. In the LHC, two beams of particles are accelerated to the ultra-relativistic energies and brought to collide. In such an environment, high energy density leads to the appearance of very rare phenomena. To understand these processes, physicists compare recorded data with accurate theoretical models simulations. Currently employed simulation techniques use complex Monte Carlo processing in order to compute all possible interactions between particles and matter. Such an approach produces accurate results at the expense of high computational cost.

Therefore multiple attempts are taken to speed up this processing, including those that leverage state of the art generative models (Paganini et al., 2018; Khattak et al., 2018; Deja et al., 2018) such as Generative Adversarial Networks (Goodfellow et al., 2014a) (GANs) or Variational Autoencoders (Kingma and Welling, 2014) (VAE). While the above methods are much faster than standard simulations, they suffer from limitations which make them unsuitable for reliable real data simulation tool. Training of Generative Adversarial Networks is often unstable and as noticed by Arora et al. (2017); Arora and Zhang (2017) it may result in limited

quantitative properties. On the other hand, Variational Autoencoders converge in steady manner. However, because of the maximum likelihood approximation they also produce blurry results with both visual and statistical problems. In this work we address those shortcomings, and propose a novel solution built on top of recent advancements in generative modelling.

In particular, we propose a new generative model build on top of the Sinkhorn Autoencoder introduced by Patrini et al. (2019). However, instead of restricting autoencoder to encode examples on the parametrised distribution, we approximate it with explicit *noise generator* implemented through the additional deterministic neural network, as presented in Fig. 4.1.1. We input noise from a known distribution (*e.g.* normal) to this network and encode it to match the distribution of real data in the autoencoder's latent space. Although such an approach allows us to generate new data samples from a parametrised distribution, thanks to an additional neural network, we do not regularise our encoder's latent space with such constraints. To our knowledge our end to end Sinkhorn Autoencoder is the first generative autoencoder without explicit constraint on the latent space.

On top of this approach, we extend our setup to conditional generative models. Currently proposed methods such as conditional VAE (condVAE) (Sohn et al., 2015) introduce additional conditioning parameters to the encoder and decoder. At the same time condVAE regularises latent space to follow normal distribution. Therefore, the model learns to encode information related to classes only in encoder and decoder, while in latent space all of the examples are shuffled into a single manifold as presented in Fig. 4.1.1. This behaviour limits classes separation, since they have to be learned in decoder from one common continuous distribution.

In this work we introduce a conditional version of our solution. Contrary to prior methods, we do not input conditional parameters into the encoder and decoder. We allow autoencoder to encode different classes in different areas of the latent space, while we match them with the conditional noise generator. Such an approach, is more suitable for different (e.g. imbalanced) conditional classes. It allows to encode data into a more natural, disentangled representations with clear classes separation as depicted in Fig. 4.1.1.

We evaluate the quality of our standard and conditional end-to-end Sinkhorn Autoencoder with commonly used benchmark datasets, such as MNIST (LeCun et al., 1998) and CelebA (Liu et al., 2015a), and achieve state-of-the-art results. To show generalisation of our solution we then apply it to the problem of fast simulation of particle showers in High Energy Physics (HEP). We show that our method allows to generate high-quality calorimeter responses from the whole distribution of original data. The superiority of our model is even more pronounced on this dataset.

The main contributions of this work are:

- A new non-adversarial end-to-end generative model with explicit noise generator.
- A novel conditional generative model based on the autoencoder architecture which, contrary to the currently employed models, leaves the structure of autoencoder's latent space intact.

The remainder of this work is organised as follows. In Sec. 6.3 we describe related works in the field of autoencoding generative modelling and fast simulations for HEP. Sec. 4.3 introduces our end-to-end Sinkhorn Autoencoder method followed by its conditional version. We conclude this work in Sec. 4.4, with experiments on MNIST, CelebA and HEP datasets and a description of potential further studies.

4.2. Related Works

In this work, we relate to the generative autoencoder setup overviewed in Chapter. 3.1. In particular, to the Sinkhorn Autoencoder (Patrini et al., 2019) we use as a basis of our method. Contrary to this work, we focus on the prior-posterior mismatch in generative autoencoders that result in low-quality image generations.

In this work, we address those limitations with our end-to-end sinkhorn autoencoder, by leveraging vanilla autoencoder as a method for learning low-dimensional data representations that are approximated by additional neural network.

4.3. Sinkhorn Autoencoder with Noise Generator

In this section, we describe our new end-to-end Sinkhorn Autoencoder generative model. We first introduce the general overview presented in Fig. 4.3.1. Then, we examine three parts of the final model optimisation objective: a reconstruction loss, a Sinkhorn loss applied on the latent space, and additional regularisations. Finally, we introduce a conditional version of our model.

In the Sinkhorn Autoencoder work Patrini et al. (2019) introduce a generative model where the Sinkhorn algorithm is used to match the autoencoder's latent space with a known distribution. In our work, we leverage this analysis and present an extended version of this method with a trainable prior approximator dubbed *noise generator* implemented as a neural network. We train the noise generator jointly with the encoder using the Sinkhorn loss applied on the latent space.

In Fig. 4.3.1 we demonstrate the general layout of our solution. It consists of three neural networks – *encoder* q_ϕ, *decoder* p_θ and *noise generator* n_ν. As presented

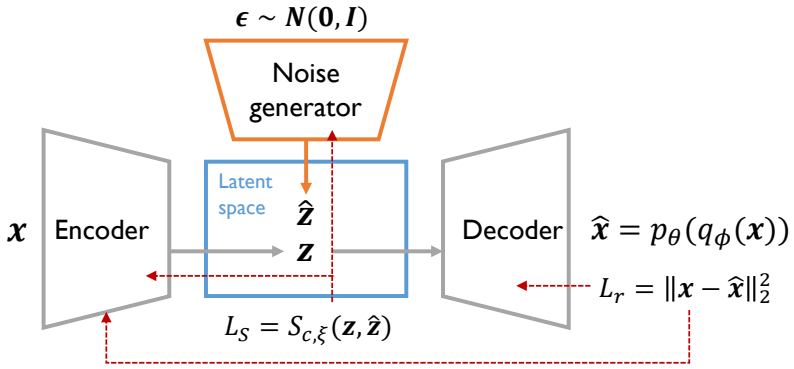


Figure 4.3.1. The architecture of the Sinkhorn Autoencoder with a neural network as an explicit noise generator. Red arrows indicate the gradient flow. Reconstruction Loss L_r is backpropagated through decoder and encoder, while Sinkhorn loss L_S is propagated in two directions to encoder and noise generator. Encoder network is optimised with a sum of L_S and L_r losses.

in Fig. 4.3.1, the loss of our model composes of two main terms - L_S - Sinkhorn loss on latent space and L_r reconstruction loss of the autoencoder. Additionally, to prevent overfitting and promote diversity, we employ regularisations on both model weights and autoencoder's latent space.

4.3.1. Reconstruction loss

The core of our network is based on a standard autoencoder. Hence, it follows the original autoencoder training procedure. The encoder is trained to map the original data $\mathbf{x}$ into the latent space $\mathbf{z}$, while the decoder is optimised to reconstruct original data examples $\hat{\mathbf{x}}$. In this part, we experiment with two different losses. Standard L_2 norm loss $L_2 = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2$ is a simple choice, but as denoted by Bojanowski et al. (2017) it may lead to blurry images. Therefore, following this work, we also employ Laplacian pyramid L_{Lap} loss presented below:

$$L_{Lap}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_j 2^{2j} |L^j(\mathbf{x}) - L^j(\hat{\mathbf{x}})|_1 \tag{4.1}$$

where $L^j(\mathbf{x})$ is the j-th level of the Laplacian pyramid representation of $\mathbf{x}$ (Ling and Okada, 2006).

Similarly to Bojanowski et al. (2017), as a final reconstruction objective we use a weighted mean of the standard mean squared error and the Lap_1 loss.

$$L_r(\mathbf{x}, \hat{\mathbf{x}}) = \alpha L_{Lap}(\mathbf{x}, \hat{\mathbf{x}}) + L_2(\mathbf{x}, \hat{\mathbf{x}}) \tag{4.2}$$

where α is a scaling parameter.

4.3.2. Sinkhorn loss

To map the generated noise onto autoencoder's latent space, we leverage Sinkhorn algorithm (Cuturi, 2013). However, contrary to Patrini et al. (2019), we use gradient obtained from this loss to optimize parameters of both our *encoder* q_ϕ and additional network – *noise generator* n_ν. We train both of those models so that their outputs – encoded data and encoded random noise – follow the same distribution in the latent space.

This proceeds as follows. First, we encode the batch of original images $\mathbf{x}$ to obtain their encoded representation $\mathbf{z} = q_\phi(\mathbf{x})$. At the same time, we process a random vector ϵ sampled from a known distribution (e. g. $\epsilon \sim \mathcal{N}(0,1)$) through the noise generator. It creates noise representation $\hat{\mathbf{z}} = n_\nu(\epsilon)$ in the same latent space as for encoded images.

To calculate the distance between noise representations $\hat{\mathbf{z}}$ and data representations $\mathbf{z}$ we use the Wasserstein distance. Following WGAN or WAE architectures, we could approximate this with an additional neural network, but to simplify the solution, we opt for entropy regularisation of the Wasserstein distance implemented with Sinkhorn algorithm.

For this purpose we follow (Genevay et al., 2017, 2018) to define the entropy regularised Optimal Transport cost with $\xi \geq 0$ as:

$$\tilde{S}_{c,\xi}(\mathbf{z},\hat{\mathbf{z}}) = \inf_{\Gamma \in \Pi(\mathbf{z},\hat{\mathbf{z}})} \mathbb{E}_{(z,\hat{z})\sim\Gamma}[c(z,\hat{z})] + \xi \cdot KL(\Gamma, \mathbf{z} \otimes \hat{\mathbf{z}}). \tag{4.3}$$

As suggested in Patrini et al. (2019) we remove the entropic bias of the above approximation with three passes of the Sinkhorn algorithm, as presented below:

$$S_{c,\xi}(\mathbf{z},\hat{\mathbf{z}}) = \tilde{S}c,\epsilon(\mathbf{z},\hat{\mathbf{z}}) - \frac{1}{2}(\tilde{S}c,\epsilon(\mathbf{z},\mathbf{z}) + \tilde{S}c,\epsilon(\hat{\mathbf{z}},\hat{\mathbf{z}})) \tag{4.4}$$

With the above equation we calculate the loss value for two representations of encoded images $\mathbf{z}$ and generated noise $\hat{\mathbf{z}}$ in a batch-wise manner as $S_{c,\xi}(\mathbf{z},\hat{\mathbf{z}})$. For the Wasserstein cost function c we use standard 2-Wasserstein distance with euclidean norm $c(a,b) = \frac{1}{2}||a-b||_2^2$. As indicated in Genevay et al. (2018) $S_{c,\xi}$ deviates from the original Wasserstein distance by approximately $O(\xi log(\frac{1}{\xi}))$, hence we keep our ξ small to avoid the influence on network's convergence. In practice, we use the efficient implementation of the Sinkhorn algorithm with GPU acceleration from GeomLoss package (Feydy et al., 2019).

4.3.3. End-to-end Sinkhorn Autoencoder objective

To improve the diversity of generated images, we include additional regularisation on the autoencoder's latent space. For this purpose, we adapt diversity regularisation proposed in Ayinde et al. (2019) . In this work, authors compute a similarity matrix SIM_C to assess the diversity in the neural network's weights according to the cosine similarity between the outputs of consecutive layers.

We adapt this technique in our model to measure the similarity between all of the encoded real data examples from the batch. Then, following Ayinde et al. (2019) we compute the regularisation as a sum of these similarities as presented in 4.5:

$$R_s(\mathbf{z}) = \rho \sum_{i=1}^{bs} \sum_{j=1, j\neq i}^{bs} m_{i,j} \left(SIM_C \left(\mathbf{z}_i, \mathbf{z}_j^T \right) \right)^2 \tag{4.5}$$

where ρ is a scaling factor bs is batch size and m is a binary mask variable which drops pairs below threshold τ.

$$m_{i,j} = \begin{cases} 1, & \left| SIM_C \left(\mathbf{z}_i, \mathbf{z}_j^T \right) \right| \geq \tau \\ 0, & otherwise \end{cases} \tag{4.6}$$

Additionally, we also experiment with different regularisations on autoencoder's weights. In our experiments, we observed better convergence with L_2 regularisation on the last layer of our encoder.

Below we outline the joint loss function of our autoencoder as a sum of four elements: reconstruction loss, Sinkhorn loss between generated noise and original encoded images, and additional regularisation on the network's latent space.

$$\begin{aligned} L = & \alpha L_r(\mathbf{x}, p_\theta(q_\phi(\mathbf{x}))) \\ & + \beta Sc, \xi((q_\phi(\mathbf{x}), n_\nu(\epsilon \sim \mathcal{N}(0,1)) \\ & + \delta R_s(q_\phi(\mathbf{x})) \end{aligned} \tag{4.7}$$

4.3.4. Conditional Sinkhorn objective

While the goal for most applications of generative modelling is to generate more examples from a given distribution, for certain tasks we have to include additional information about simulated data. This is also the case for HEP, where we want to simulate possible responses of a calorimeter for a given particle. For the purpose of conditional images generation, we propose a simple adjustment to the standard version of our method.

Firstly, for a given batch of samples $\mathbf{x}$ with corresponding conditioning variables

$\mathbf{y}$, we propose to pass the original conditional values to the *noise generator* as a separate input for the neural network. Thanks to this, we change our *noise generator* to encode random noise ϵ with respect to conditional values $\mathbf{y}$ as $\hat{\mathbf{z}} = n_v(\epsilon, \mathbf{y})$.

Secondly, we train the *noise generator* and the *encoder* to encode examples with similar conditional values near to each other in the latent space. For that purpose, we first encode all of the examples $\mathbf{x}$ into their representation in the latent space $\mathbf{z}$. Then, for each example $z \in \mathbf{z}$, we concatenate its encoding with corresponding conditional values $y \in \mathbf{y}$. We perform the same operation for noise latent representations $\hat{\mathbf{z}}$ and the same conditional parameters $\mathbf{y}$ from original training data. Finally, we pass concatenated vectors through the Sinkhorn algorithm, calculating the loss value.

$$S_{c,\xi}(q_\phi(\mathbf{x}) \oplus \mathbf{y}, n_v(\epsilon \sim \mathcal{N}(0,1), \mathbf{y}) \oplus \mathbf{y}) \tag{4.8}$$

, where $\oplus$ denotes vectors concatenation.

This approach aligns original data latent representations $\mathbf{z}$ for conditional values $\mathbf{y}$ with a noise embeddings $\hat{\mathbf{z}}$ conditioned on the same values $\mathbf{y}$. For the *Wasserstein*$_2$ metric the associated part of the loss fucntion value is equal to the euclidean norm distance between different conditional values. However, depending on the nature of the conditional information $\mathbf{y}$ and the potential real cost of generating samples from the distribution related to other conditions, it might be beneficial to scale it accordingly.

With such an approach, contrary to the other conditional generative models such as conditional VAE or WAE, our solution leaves the original autoencoder's latent space intact. We do not enforce it to encode different classes into the single normal distribution. Thanks to the fact that conditional parameters are included in the noise generator, we can observe that autoencoder distribute different classes in separate areas of its latent space. We compare both exemplar latent spaces for the MNIST dataset in figure Fig. 4.1.1.

4.4. Experiments

In this section, we evaluate the performance of our end-to-end Sinkhorn Autoencoder model in reference to other generative solutions. Primarily, we compare the results of our experiments to other autoencoder based generative approaches. For that purpose, we use two standard benchmarks: the MNIST dataset of handwritten digits LeCun et al. (1998) and CelebA dataset of celebrity face images Liu et al. (2015a), cropped to 64x64 pixels. We also evaluate different conditional genera-

Table 4.3.1. Results comparison for conditional generative models on MNIST and HEP datasets. Our conditional e2eSAE outperforms other conditional methods on both standard benchmark as well as HEP dataset. In terms of the latter our model is able to simulate the exact localisation of collision with very high accuracy.

	MNIST		HEP	
Model	FID	Sinkhorn	MAE	Sinkhorn
cond VAE	6.61	30.13	23.13 (±65.53)	14.92
cond WAE (MMD)	34.73	30.44	43.54 (±55.23)	34.46
cond e2eSAE (ours)	**4.11**	**24.92**	**13.50 (±29.82)**	**7.91**
cond DCGAN	0.93	22.23	68.27 (±180.45)	6.95
original data	0.33	0	6.59	2.89

tive solutions, including the adversarial example of conditional Deep Convolutional GAN Radford et al. (2015) on a challenging dataset of calorimeter response simulations.

The dataset for the latter, which we call HEP, consists of 117 817 Zero Degree Calorimeter responses to colliding particles, calculated with the full GEANT4 Incerti et al. (2018) simulation tool. Each particle response simulation starts with the single particle described with 9 attributes (mass, momenta, charge, energy, primary vertex). Particle is then propagated through the detector where simulation tools are employed to calculate all of its interactions with the detector's matter. The final outcome of a simulation is the result of those interactions observed as a total energy deposited in calorimeter's fibres. Since those fibres are arranged in a grid with 44×44 size, we can treat the final response as an image with 44×44 pixels. Visualisation of such simulations is presented in Fig. 4.4.1. Although, resulting images are non-deterministic, they are highly affected by initial particle attributes. In principle, particle type (mass and charge) defines the trajectory of particle, while its energy and momenta directly influence the luminosity of the response.

In our experiments, we use network architectures proposed in Wasserstein Autoencoder Tolstikhin et al. (2017). Therefore for the CelebA dataset, we use a convolutional deep neural network with 4 convolutional/deconvolutional layers for both encoder and decoder with 5x5 filters. Additionally we use batch normalisation Ioffe and Szegedy (2015) after each convolutional layer. For our noise generator, we use a simple, fully connected network with 3 hidden layers and ReLU activations. We optimise our networks with Adam Kingma and Ba (2014) in batches of 1000 examples.[1]

To assess the quality of generated samples we use *Fréchet inception distance*

[1] Code for our work is available at `https://github.com/KamilDeja/e2e_sinkhorn_autoencoder`

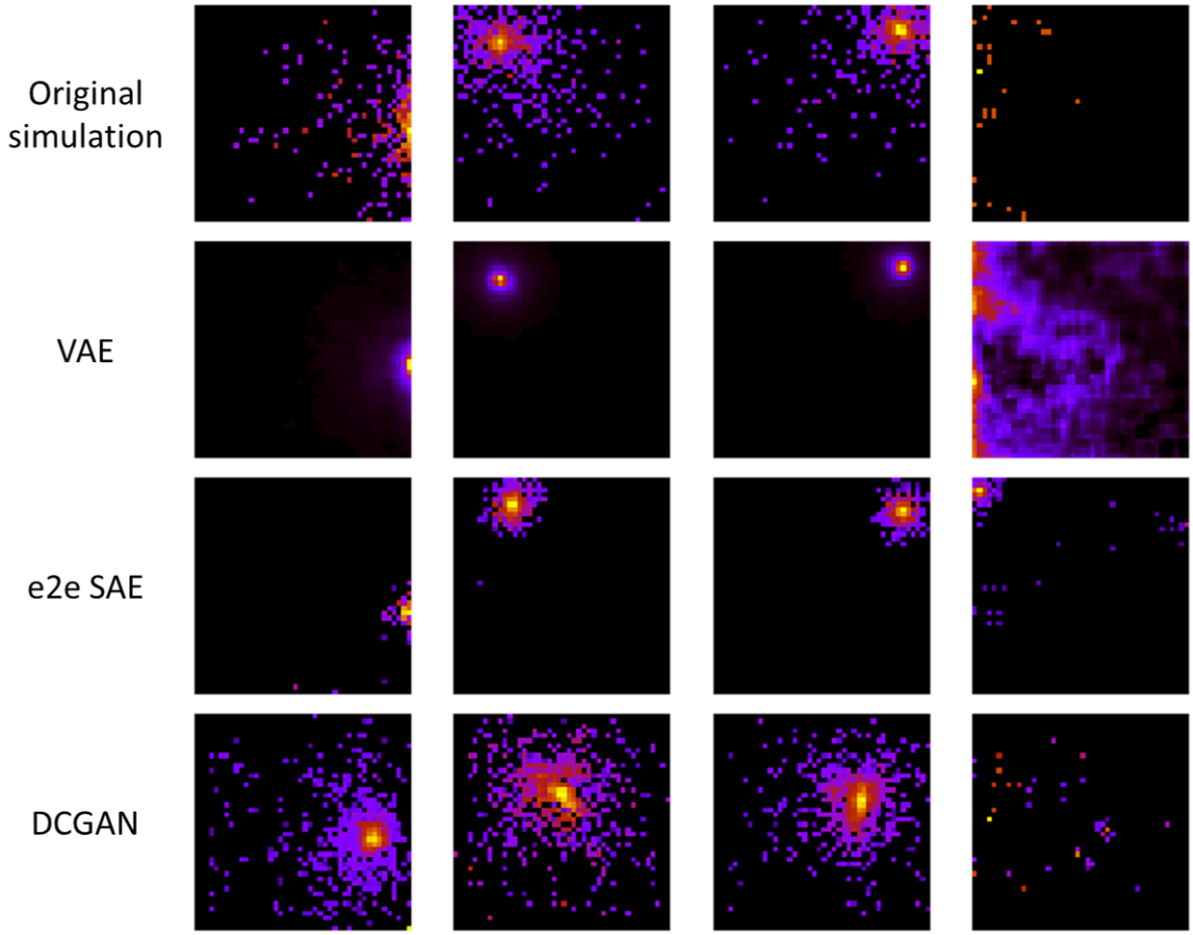
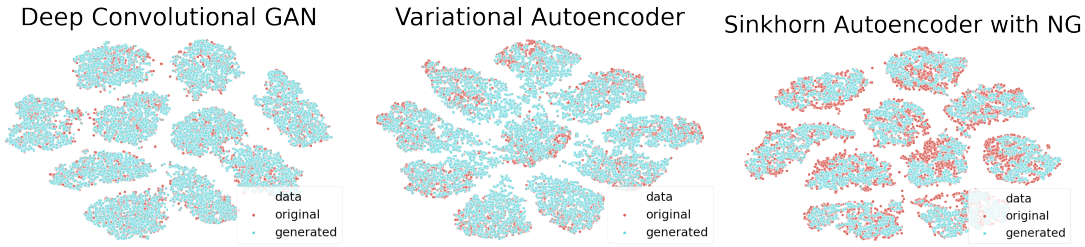


Figure 4.4.1. Examples of calorimeters response simulations with different methods. Although results from GAN are visually sound with collisions, model was not able to properly capture relations from conditional values. Our solution does not reproduce all of residual values, but it outperforms other methods in terms of accuracy of positioning for the most significant centre of the collision.

(FID) introduced in Heusel et al. (2017). As proposed in Bińkowski et al. (2018), for the MNIST dataset, we change the original Inception neural network to the LeNet based convolutional classifier. While FID is criticised for approximating distributions with Gaussians Shmelkov et al. (2018), we also introduce a new measure to monitor the diversity of generated examples. After propagating original and generated images up to the LeNet's penultimate layer, we compare their distributions with Wasserstein distance approximation implemented with Sinkhorn algorithm. We refer to this measurement as *Sinkhorn*.

For the HEP dataset, we benefit from the fact that the original data is simulated, hence we can assess the quality of generated samples on the basis of their physical properties. Following the calorimeter's specification Dellacasa et al. (1999), we sum pixels of generated images into five *channels*. Calculated channels are usually employed for the calibration purposes, since they represent well the physical properties



(a) **(b)** **(c)**

Figure 4.4.2. T-SNE visualisation of generated examples (blue) and original mnist data (red), processed through the LeNet model. Well trained DCGAN, without mode collapse, reproduces the whole data distribution well, while VAE additionally produces images from outside of real data distribution (between real classes). Our solution (right) generates only examples within true data distribution but has minor problems with reproducing their whole variety.

Table 4.4.1. Results comparison on the CelebA dataset. For competitive solutions we include the best of reported result. Our solution outperforms recent non-adversarial generative models.

CelebA	
Model	FID
VAE + Flow	65.7
SWAE	64
SAE (H)	56
VAE	55
WAE (MMD)	55
e2e SAE (ours)	**54.5**

of simulated collision. To assess the quality of generations we compare them to the original full simulations by measuring mean absolute error between channels from original and generated responses with the same input. In table 4.3.1 we refer to this measurement as MAE. While original simulations are also non-deterministic and there are several realistic outputs for the same particle, their general characteristic captured by channels deviates by a small margin of 6.59 MAE. This error allows us to measure how accurate our model is in terms of data generation with respect to conditional values. To measure how well it reproduces the whole distribution of channel values, we also calculate the Wasserstein distance between original and generated channels distribution on the whole test-set.

As presented in Tab. 4.3.1, our conditional model outperforms other non--adversarial solutions on both MNIST and HEP datasets. As shown in Fig. 4.4.1, HEP dataset remains challenging for all generative models. For VAE, we can see the blurry generations as an outcome of regularisation with the normal distribu-

Figure 4.4.3. Samples of generated images from model trained on CelebA dataset. Our model is capable of generating diverse, high quality images without blurred effect.

tion. Thanks to the adversarial training with discriminator instead of averaged reconstruction error DCGAN produces visually more attractive results. However, our end-to-end Sinkhorn Autoencoder with Noise Generator better captures relations between conditional parameters such as the centre of collision, what is of the special interest in real data simulation.

For MNIST, we also visually analyse coverage of data distribution for evaluated methods. As presented in T-SNE visualisation of LeNet penultimate layer activations in Fig. 4.4.2, our method has better coverage of original data than other autoencoder based approaches. As depicted in Fig. 4.4.2c, our model does not produce examples outside of original data distribution. On the other hand, results from the well-trained adversarial model (Fig. 4.4.2a) better overlaps with the full data distribution of the MNIST dataset.

On the CelebA dataset, as demonstrated in table 4.4.1, our method outperforms other competitive autoencoder based solutions. As displayed in figure Fig. 4.4.3, our end-to-end Sinkhorn Autoencoder generates visually sharp images with high variance.

4.5. Conclusions

In this work, we introduced a new generative model based on autoencoder architecture. Contrary to contemporary solutions, our end-to-end Sinkhorn Autoencoder does not enforce encoding on any parametrised distribution. In order to learn the distribution of standard autoencoder, we converge to it with an additional deterministic neural network, which we train together with an autoencoder. We show that our solution outperforms other comparable approaches on benchmark datasets and the challenging practical dataset of calorimeter response simulations. We postulate that the general approach proposed in this work may also be used with other metrics between probability distribution.

5. On Analyzing Generative and Denoising Capabilities of Diffusion-based Deep Generative Models

Title	On Analyzing Generative and Denoising Capabilities of Diffusion-based Deep Generative Models
Authors	Kamil Deja, Anna Kuzina, Tomasz Trzciński, Jakub M. Tomczak
Conference	Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS22)
Year	2022

Preface

So far, we have focused on one family of generative models known as generative autoencoders, where data examples are encoded into the latent representation $\mathbf{z}$ and decoded back with a marginal distribution $p(x|z)$ with a single decoder. In such an approach, data samples are encoded into representations with a single pass through a neural encoder that maps the input into independent latent variables. While it simplifies the process, the encoded features are often complex and hard to understand. In Sec. 3.1, we describe hierarchical approaches, where latent features are encoded one by one so that previous latent factors influence the future ones. In this setup, learned latent features are more informative and can capture more complex relationships between different data characteristics. As a result, hierarchical VAEs can model a broader range of abstractions in the data by representing them at different levels of granularity.

Those possibilities are further explored in deep generative diffusion models (DDGMs). In this technique, images are "encoded" with a forward diffusion process, where a Gaussian noise from a pre-defined distribution is iteratively added to distort training data. New instances are generated in a backward diffusion process that inverts the forward one. Specifically, a single decoder is applied numerous times to convert randomly sampled noise into image features gradually. Although there is no notion of latent space with hidden data features in such formulation, in this chapter, we analyse the temporary working image updated in consecutive diffusion steps.

Therefore, we focus on the structure of data representations in the DDGMs. In the series of analysis, we show that those models can be roughly divided into two parts based on the diffusion process. The first one generates new data features similar to the one observed in the training data, while the second removes the remaining noise in a data-agnostic way. On top of this analysis, we introduce a hybrid model, where we first use a DDGM denoiser to generate new data features, followed by a deterministic denoising auto-encoder (DAE) that removes the remaining noise. In this setup, we use the first model to learn data features from the training data probability distribution, while the DAE is used only to improve image quality.

Abstract

Diffusion-based Deep Generative Models (DDGMs) offer state-of-the-art performance in generative modelling. Their main strength comes from their unique setup in which a model (the backward diffusion process) is trained to reverse the forward diffusion process, which gradually adds noise to the input signal. Although DDGMs are well studied, it is still unclear how the small amount of noise is transformed during the backward diffusion process. Here, we focus on analysing this problem to gain more insight into the behaviour of DDGMs and their denoising and generative capabilities. We observe a fluid transition point that changes the functionality of the backward diffusion process from generating a (corrupted) image from noise to denoising the corrupted image to the final sample. Based on this observation, we postulate to divide a DDGM into two parts: a denoiser and a generator. The denoiser could be parameterised by a denoising auto-encoder, while the generator is a diffusion-based model with its own set of parameters. We experimentally validate our proposition, showing its pros and cons.

5.1. Introduction

Diffusion-based Deep Generative Models (Sohl-Dickstein et al., 2015) (DDGM) have recently attracted increasing attention, due to the unprecedented quality of generated samples (Dhariwal and Nichol, 2021; Ho et al., 2022; Kingma et al., 2021). The general idea behind this set of methods is to generate samples using diffusion processes (Ho et al., 2020; Huang et al., 2021; Kingma et al., 2021; Song and Ermon, 2019; Song et al., 2020b). In the *forward diffusion process*, an image is passed through a number of steps that consecutively add a small portion of noise to it. The *backward diffusion process* is a direct reverse of the forward process, where a generative model is trained to gradually denoise the image. With a sufficient number of the forward diffusion steps, noisy images approach isotropic Gaussian noise. Then, generating new examples is possible by applying the backward diffusion to the noise sampled from the standard Gaussian distribution.

While the performance of DDGMs is impressive, not all of their aspects are fully understood. Intuitively, a DDGM is trained to *remove* small amounts of noise from many intermediary corrupted images. Although this perspective is reasonable and complies with the interpretation of DDGMs using stochastic differential equations (Huang et al., 2021; Song et al., 2020b), it is still unclear how the small amount of noise is *removed* during the backward diffusion process where images are composed of almost entirely random values. The more adequate intuition might be that in its

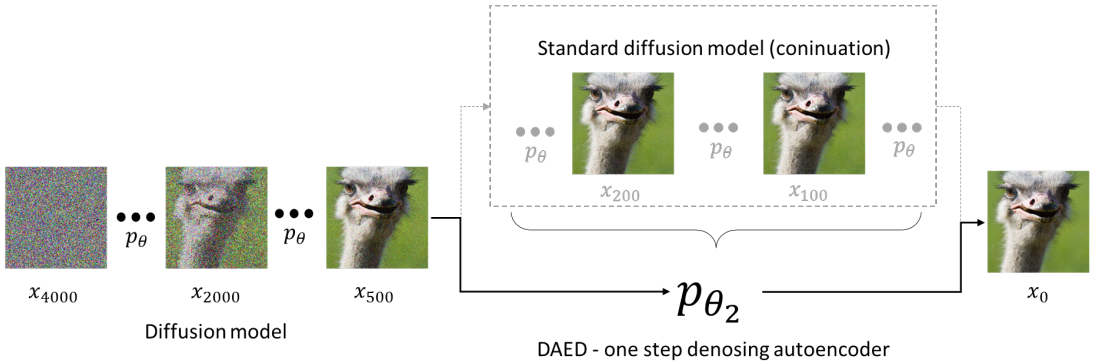


Figure 5.1.1. Overview of the proposed Denoising Auto-Encoder with Diffusion (DAED). To validate our hypothesis that DDGMs can be understood as a composition of a *generator* and *denoiser*, we propose to explicitly model the denoising part with a separate denoising autoencoder.

initial steps, a diffusion model does not only remove noise but also introduces a new signal according to the distribution learned from the data. In this work, we further investigate this observation to understand the balance between the generative and denoising capabilities of DDGMs.

In particular, we aim to answer the following three questions in this paper: **(i)** Is there a transition in the functionality of the backward diffusion process that switches from generating to denoising? **(ii)** How does this split of functionality affect the performance? **(iii)** Does the denoising part in DDGMs generalize to other data distributions? As a result, the contribution of the paper is threefold:

- First, we analyze the noise distribution in the forward diffusion process and how steps of the diffusion process are correlated with the reconstruction error.
- Second, based on our analysis, we postulate that DDGMs are composed of two parts: a *denoiser* and a *generator*. As a result, we propose a new class of models that consist of a Denoising Auto-Encoder and a Diffusion-based generator shortened as DAED. DAED could be considered as a variation of DDGMs with an explicit split into the denoising part and the generating part.
- Third, we empirically assess the performance of DDGMs and DAED on three datasets (FashionMNIST, CIFAR10, CelebA) in terms of data generation and transferability (i.e., how DDGMs behave on different data distribution).

5.2. Background

In this work, we analyse Diffusion-Based Deep Generative Models. We follow their formulation as introduced by Ho et al. (2020); Sohl-Dickstein et al. (2015) and described in Chapter 2.3.

5.3. Denoising Auto-Encoders

Another class of models, Denoising Auto-Encoders (DAEs), is similar to DDGMs in the sense that they also revert a known corruption process. However, DAEs are trained to remove the noise in a single pass, and unlike DDGMs, they cannot generate new objects. Specifically, DAEs are auto-encoders that reconstruct a data point $\mathbf{x}_0$ from its corrupted (noisy) version (Alain and Bengio, 2014; Bengio et al., 2013; Chen et al., 2014; Vincent et al., 2008). Let us denote the auto-encoder by $f_\varphi(\cdot)$. Using the same notation as for DDGMs, the Gaussian corruption distribution is $q(\mathbf{x}_1|\mathbf{x}_0)$. Then, a DAE maximizes the following objective function:

$$\ell(\mathbf{x}_0;\varphi) = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\left[\ln p\left(\mathbf{x}_0|f_\varphi(\mathbf{x}_1)\right)\right]. \tag{5.1}$$

and, in particular, for the Gaussian distribution with the identity covariance matrix, we get the original objective for DAEs Vincent et al. (2008): $\ln p\left(\mathbf{x}_0|f_\varphi(\mathbf{x}_1)\right) = -\left\|\mathbf{x}_0 - f_\varphi(\mathbf{x}_1)\right\|^2 + const.$

5.4. Related Works

Various modifications of DDGMs were recently proposed to improve their sampling quality. This includes simplifying the learning objective and proposing new noise schedulers, which allow DDGMs to achieve state-of-the-art results (Nichol and Dhariwal, 2021; Dhariwal and Nichol, 2021). In this work, we show that splitting the decoder into two parts, namely, a denoiser and a generator, can benefit the performance, especially when training with the variational lower bound.

Properties of DDGMs Ho et al. (2020) notice that DDGMs can be beneficial for lossy compression, observing (Figure 5 in (Ho et al., 2020)) that most of the bits are allocated to the region of the smallest distortion that corresponds to the first steps of a DDGM. We draw a similar conclusion when discussing the denoising ability of the diffusion model in Section 5.5. However, we base our analysis on the signal-to-noise ratio rather than compression. On the other hand Salimans and Ho (2022) focus on the computational complexity of DDGM and propose a progressive distillation that iteratively reduces the number of diffusion steps. The work shows that it is possible to considerably reduce the number of sampling steps without losing performance. We believe that their results support our intuition that it is reasonable to combine several initial steps into a single denoiser model. Benny and Wolf (2022) evaluate how the diffusion process changes in time when model is trained with different objectives (Eq. 2.12 or Eq. 2.13). They observe that the

image generation process differs significantly and that it is more beneficial to switch between those two approaches at different stages of the diffusion. As mentioned in Chapter 3.2 there are several works that draw connection between diffusion models and hierarchical VAEs. In this work, we follow a similar direction and propose to see a DDGM as a combination of a denoising auto-encoder with an additional diffusion-based generator of corrupted images.

5.5. An Analysis of DDGMs

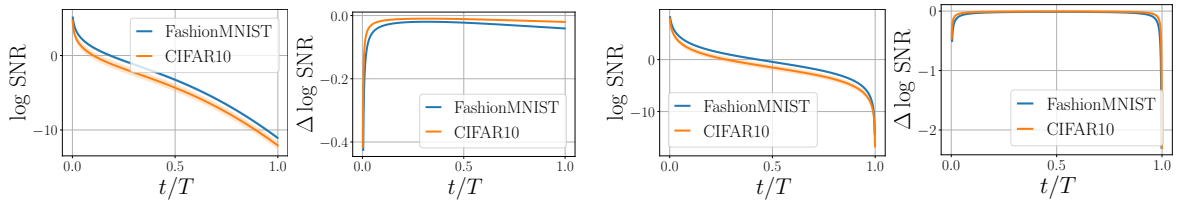
The core idea behind DDGMs is the gradual noise injection to images as we go forward in time such that the final object is a sample from the standard Gaussian distribution. Then, in the backward diffusion process model reverts this procedure and, as a result, generates new objects. Therefore, understanding the success of DDGMs relies heavily on understanding how the injected noise influences the behavior of both training and the model itself.

The noise distribution in the forward diffusion process The first question we ask is how much corrupted an image gets after applying a specific noise schedule. Following Ho et al. (2020); Kingma and Welling (2014); Nichol and Dhariwal (2021), we can utilize the signal-to-noise ratio (SNR), expressed as the squared mean of a signal (here: image) divided by the variance of a signal, to quantify the amount of noise in $\mathbf{x}_t$. For this purpose, the quantity of interest is the forward diffusion for a given $\mathbf{x}_0$, namely, $q(\mathbf{x}_t|\mathbf{x}_0)$, that results in the following SNR:

$$SNR(\mathbf{x}_0, t) = \frac{\overline{\alpha}_t \mathbf{x}_0^2}{1 - \overline{\alpha}_t}. \tag{5.2}$$

Similarly to Kingma et al. (2021), we formulate the forward diffusion in such a way that the SNR is strictly monotonically decreasing in time, namely, $SNR(\mathbf{x}_0, t) < SNR(\mathbf{x}_0, s)$ for $t > s$. This means that an image becomes more noisy as we go forward in time.

In Figure 5.5.1 (left) we plot the logarithm of the SNR for both linear (Figure 5.5.1.a) and cosine (Figure 5.5.1.b) noise schedules for two datasets (FashionMNIST and CIFAR10). We average SNR over the $\mathbf{x}_0$'s (from the corresponding dataset). The right column depicts the change of the log SNR, i.e., its discrete derivative $\Delta \log \text{SNR}(t) = \log \text{SNR}(x_0, t) - \log \text{SNR}(x_0, t-1)$. First of all, we can notice a point at which the log-SNR drops below 0. This corresponds to the situation of the noise overshadowing the signal. In the case of the linear noise schedule, this happens after about 20% of steps, while for the cosine noise schedule, it appears after about



(a) Linear noise schedule **(b)** Cosine noise schedule

Figure 5.5.1. Logarithm of the signal-to-noise ratio averaged over the dataset (solid line) and its standard deviation, and the difference of the log SNR within two consecutive time steps.

25 – 50% of steps. However, the transition occurs in both cases. The biggest changes in the log-SNR are noticeable within the first 10% of steps. This may suggest that the signal is the strongest within the first 10 – 20% of the forward diffusion process steps, and then it starts being overshadowed by the noise.

The reconstruction error of DDGMs Since we know that the signal is not lost within the first 10 – 20% of steps, the next question is about the reconstruction capabilities of DDGMs, namely, what is the reconstruction error of $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$. To be clear, we are not interested in how much each step of a DDGM contributes to the final objective (e.g., see Figure 2 in (Nichol and Dhariwal, 2021)) but rather how well a DDGM reconstructs a noisy image $\mathbf{x}_t$.

In Figure 5.5.2 we plot the *Mean Absolute Error* (MAE) and the *Multi-Scale Structural Similarity* (MS-SSIM) (Wang et al., 2003) that both measure the difference between an original image $\mathbf{x}_0$ and a corrupted image at the t^{th} step $\mathbf{x}_t$ reversed by the backward diffusion. We present the values on two datasets (FashionMNIST and CIFAR10) for the first 20% of steps. Apparently, after around 10% of the steps, the reconstruction error starts growing, and the MAE increases linearly above 0.1 (i.e., about 6% of error per pixel). At the same time, the MS-SSIM drops below 0.9 – 0.95 (i.e., the discrepancy between original images and reconstructions becomes perceptually evident). This observation might suggest that DDGMs could be roughly divided into two parts: a fraction of steps of a DDGM (e.g., first 10% of the steps) constitute a *denoiser* that turns a corrupted image into a clear image, and the remaining steps of the DDGM are responsible for turning noise into a noisy structure (a corrupted

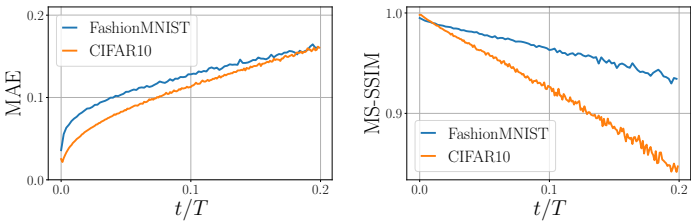


Figure 5.5.2. The averaged reconstruction error calculated using (*left*) the MAE, and (*right*) the MS-SSIM at different steps of a DDGM.

image), i.e., a *generator* that generates meaningful patterns. In other words, we claim that DDGM can be interpreted as a composition of a denoiser and a generator, but the boundary between those two parts is fluid. Moreover, the denoiser gradually removes the noise in a generative manner (i.e., by sampling $\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{x}_t)$).

DDGMs as hierarchical VAEs In this paper, we postulate that DDGMs could be seen as a composition of parts that serve different purposes. We can get additional insight into our claim by noticing a close connection between DDGMs and hierarchical VAEs. As presented by Huang et al. (2021); Kingma et al. (2021); Tomczak (2022), if we treat all $\mathbf{x}_t$'s with $t > 0$ as latents, and see the forward diffusion process as a composition of (non-trainable) variational posteriors, DGGMs become a specific formulation of hierarchical VAEs. On the other hand, we can start with a VAE with a single latent variable, $\mathbf{x}_1$, for which the variational lower bound is equal to:

$$\ln p(\mathbf{x}_0) \geq \mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1|\mathbf{x}_0)}[\ln p(\mathbf{x}_0|\mathbf{x}_1)] - D_{\mathrm{KL}}[q(\mathbf{x}_1|\mathbf{x}_0)||p(\mathbf{x}_1)]. \tag{5.3}$$

Then, similarly to Vahdat et al. (2021); Wehenkel and Louppe (2021), the marginal $p(\mathbf{x}_1)$ could be further modeled by a DDGM. By keeping the dimensionality of $\mathbf{x}_1$ the same as $\mathbf{x}_0$, and taking the variational posterior $q(\mathbf{x}_1|\mathbf{x}_0)$ to be fixed and part of the forward diffusion, we get the DDGM model. This perspective of combining a VAE with a DDGM opens new possibilities for developing hybrid models.

5.6. DAED: Denoising Auto-Encoder with Diffusion

In this work, we propose a specific combination that distinctly splits the DDGM into generative and denoising parts. As noted in the previous section, the signal in the forward diffusion process is the strongest within the first $10-20\%$ of steps, and, thus, we postulate to perceive this first part of a DDGM as a denoiser. Together with the observation about the combination of a VAE with a DDGM-based prior, we consider turning a denoising auto-encoder into a generative model as presented in Figure 5.1.1. We bring a DDGM-based part into DAE for generating corrupted images. The resulting objective is the following:

$$\overline{\ell}(\mathbf{x}_0;\varphi,\theta) = \mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1|\mathbf{x}_0)}\left[\ln p\left(\mathbf{x}_0|f_\varphi(\mathbf{x}_1)\right) + \ln p_\theta(\mathbf{x}_1)\right] \tag{5.4}$$

$$\geq \underbrace{\mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1|\mathbf{x}_0)}\left[\ln p\left(\mathbf{x}_0|f_\varphi(\mathbf{x}_1)\right)\right]}_{\ell_{\mathrm{DAE}}(\mathbf{x}_0;\varphi)} + \underbrace{\mathbb{E}_{q(\mathbf{x}_2,\ldots,\mathbf{x}_T|\mathbf{x}_1)}\left[\frac{\ln p_\theta(\mathbf{x}_1,\ldots,\mathbf{x}_T)}{q(\mathbf{x}_1,\ldots,\mathbf{x}_T|\mathbf{x}_0)}\right]}_{\ell_{\mathrm{D}}(\mathbf{x}_0;\theta)}, \tag{5.5}$$

where in (5.5) we introduce additional latent variables and the variational posterior over them, that yields the variational lower bound. We call the resulting model *DAE with a Diffusion*, or DEAD for short. In a sense, DAED is a DDGM with distinct parameterizations of the part between $\mathbf{x}_0$ and $\mathbf{x}_1$, and the part for the remaining $\mathbf{x}$'s. Thus, DEAD is almost identical to a DDGM, but there are the following differences: **(i)** We can control the amount of noise in $q(\mathbf{x}_1|\mathbf{x}_0)$. It can correspond to the first step of the forward diffusion model, or we can introduce more noise at once that would correspond to several steps in the DDGM. **(ii)** We use two different parameterizations, namely, an auto-encoder (e.g., a U-Net architecture) for $f_\varphi(\cdot)$ and a separate, shared U-Net for modeling the DDGM from $\mathbf{x}_1$ to $\mathbf{x}_T$. Since there are two neural networks, the lower bound to the objective $\overline{\ell}$ is in fact a composition of two objectives with disjunctive parameters, namely, the objective for the *denoiser*, ℓ_{DAE}, and the objective for the *generator* (i.e., the diffusion-based generative model), ℓ_{D}. **(iii)** In the DAED, we introduce the *denoiser* explicitly and make a clear distinction between the denoising and the generating parts while, as discussed earlier, this boundary is rather fluid in DDGMs. By introducing DAED, we can analyze what happens if we distinctly divide those two aspects with two separate parametrizations.

Moreover, we hypothesize that the resulting model may better generalize across various data distributions due to decoupling the parameterization of the denoiser and the generator. The training dataset may bias a single, shared parameterization in a DDGM, and while denoising an image from a different domain, it may add some artifacts from the source. While with two distinct parameterizations, there might be a lower chance for that. We evaluate this hypothesis in the experiments.

5.7. Experiments

Experimental setup In all the experiments, we use a U-Net-based architecture with timestep embeddings as proposed by Ho et al. (2020); Nichol and Dhariwal (2021). We train all the models with a linear β scheduler and uniform steps sampler to simplify the comparison. All implementation details and hyperparameters are included in the Appendix (5.9.5) and code repository [1]. For DAED, we use the same architecture for both the diffusion part and the denoising autoencoder. We run experiments on three standard benchmarks with different complexity: FashionMNIST (Xiao et al., 2017) of gray-scale 28×28 images, CIFAR-10 (Krizhevsky et al., 2009) of 32×32 natural images, and CelebA (Liu et al., 2015b) of 64×64 photographs of faces. We do not use any augmentations during training for any dataset. We report results for both variational lower bound loss (VLB) (Sohl-Dickstein et al., 2015)

[1] `https://github.com/KamilDeja/analysing_ddgm`

and simplified objective (Ho et al., 2020). Following Nichol and Dhariwal (2021) we evaluate the quality of generations with Fréchet Inception Distance (FID) (Heusel et al., 2017) and distributions Precision (Prec) and Recall (Rec) metrics (Sajjadi et al., 2018) that disentangle FID score into two aspects: the quality of generated results (Precision) and their diversity (Recall).

5.7.1. Is there a transition in functionality of the backward diffusion process that switches from generating to denoising?

In section 5.5, we investigate how the signal-to-noise ratio and the reconstruction error of a DDGM change with the increasing number of diffusion steps (see Figure 5.7.1). Based on this analysis, we postulate that DDGMs can be divided into two parts: a *denoiser* and a *generator*. To determine the switching point, we propose an experiment that answers the following question:

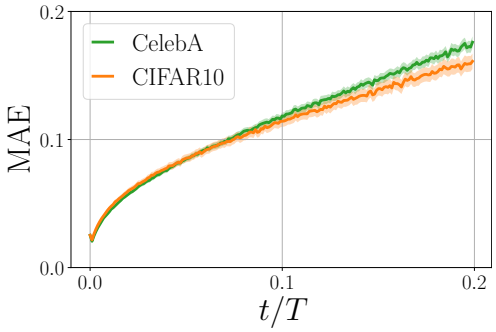


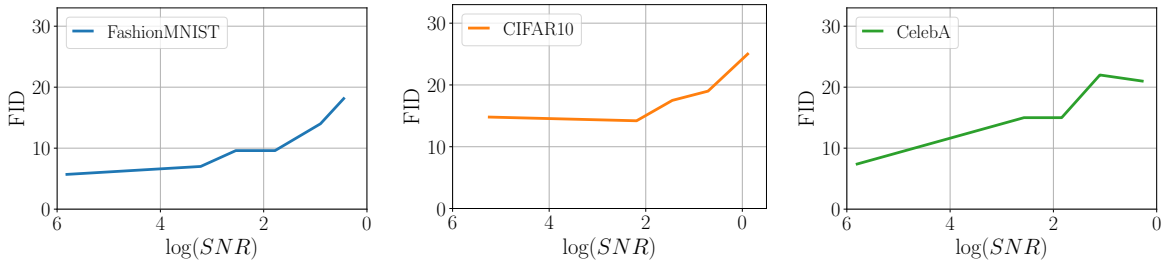
Figure 5.7.1. The MAE for a DDGM trained on CIFAR10 and evaluated on CIFAR10 & CelebA, with a 0.95 confidence interval.

Is there a denoising part of a DDGM that is agnostic to the signal from the data?

To that end, we refer once more to the analysis of the reconstruction error (e.g., MAE) from different diffusion steps. This time, however, we compare the quality of reconstructions with a single DDGM model trained on the CIFAR10 dataset and then evaluated on CIFAR10 and CelebA. The result of this experiment is presented in Figure 5.7.1. Interestingly, we notice that for approximately 10% of the initial steps of the DDGM, there is a negligible difference in the reconstruction error between these two datasets. This fact may suggest that, indeed, the model does not require any information about the background data signal in the first steps, and it is capable of denoising corrupted images. However, after this point (about 10% of steps), the reconstruction error starts growing faster for the dataset the model was not trained on. This indicates that information about the domain becomes important and affects performance.

5.7.2. How does splitting DDGMs into generative and denoising parts affect the performance?

The results so far confirm our claims that DDGMs could be divided into denoising and generative parts. Independently of a dataset, there appears to be a tran-



(a) FashionMNIST (b) CIFAR10 (c) CelebA

Figure 5.7.2. The performance (FID) of DAED with different switching points with respect to the logarithm of the signal to noise ratio (5.2) on three different datasets.

Table 5.7.1. FID Precision (Prec) and Recall (Rec) scores. For each row, we indicate the length of the diffusion process (T) and the training objective (Loss). **Best results in bold**.

Model		Fashion Mnist				CIFAR10				CelebA			
	Loss	T	FID ↓	Prec ↑	Rec ↑	T	FID ↓	Prec ↑	Rec ↑	T	FID ↓	Prec ↑	Rec ↑
DDGM	VLB	500	8.9	68	53	1000	26	53	54	1000	23	51	21
DAED $\beta_1 = 0.1$	VLB	468	9.1	**71**	60	900	20	59	46	900	18	63	**30**
DAED $\beta_1 = 0.001$	VLB	499	**7.5**	**71**	**64**	999	**15**	**60**	**60**	999	**16**	**70**	27
DDGM	Simple	500	7.8	72	**65**	1000	**7.2**	**65**	**61**	1000	**4.9**	66	**57**
DAED $\beta_1 = 0.1$	Simple	468	9.6	**73**	58	900	19	62	50	900	22	**67**	27
DAED $\beta_1 = 0.001$	Simple	499	**5.7**	69	64	999	14.8	**65**	53	999	7.4	**67**	54

sition point at which a DDGM stops generating a corrupted image from noise and starts denoising it in a generative manner. Here, we aim to verify whether it is possible to do a clear split into a denoising part and a generating part. For this purpose, we use the introduced DAED approach that consists of a DAE part (the denoiser) and a DDGM (the generator) parameterized by two distinct U-Nets.

First, we consider a situation in which we train a DDGM using the simplified objective (2.13) and then replace the first steps with a DAE. In other words, we train a DAED in two steps: first the DDGM and then the DAE. This experiment aims to check how the decoupling of the DDGM into two parts influences the model performance. In Figure 5.7.2 we present the dependency between the log-SNR at the splitting point and the FID score. In all cases, the performance of DAED is comparable to the DDGM if we replace the DAE with up to the 10% of the steps that correspond to $log(SNR)$ is equal to around 4. For more complicated datasets like CIFAR10 and CelebA, fewer steps could be replaced. This effect could be explained by the fact that images in these datasets have three channels (RGB), and removing noise is more problematic. That outcome reconfirms our presumptions that it is reasonable to split the DDGM since the final performance is not significantly affected by the division for an adequately chosen splitting point.

To get further insight into the qualitative performance, in Figure 5.7.3 we demonstrate how the selection of the splitting point with respect to the Signal to Noise Ratio (SNR) affects the quality of final generations[2]. We present non-cherry-picked samples from DAED trained in the same manner as described in the previous paragraph. As expected, the more noise the DAE part (the denoiser) must deal with (see the values of β_1 in Figure 5.7.3), the fewer details in the generations there are. These samples again indicate that by replacing some steps with a denoiser, we get a trade-off between "cleaning" the corrupted image or, in fact, further generating details. It seems that there is a sweet spot for perceptually appealing images that contain details and are "smooth" at the same time, see $\beta_1 = 0.025$ in Figure 5.7.3. However, as it is typically difficult to provide convincing arguments by *staring* at samples, we further propose to analyze quantitative measures.

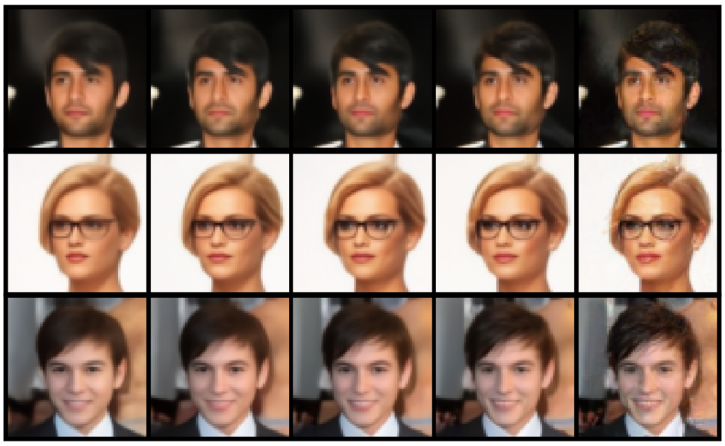


Figure 5.7.3. Examples of generations from DAED with the same noise value and different switching points.

In Table 5.7.1, we compare the performance of DAED against the DDGM on FashionMNIST, CIFAR10, and CelebA in terms of FID, Precision and Recall scores. We want to highlight that our goal is not to achieve SOTA results on the before-mentioned datasets but to verify whether we can gain some further understanding and, potentially, some improvement by splitting the denoising and generative parts. We consider two scenarios, namely, learning a DDGM and DAEDs using either the variational lower bound (VBL) or the simplified objective (Simple) with various lengths of the diffusion. Interestingly, DAED outperforms the DDGM when these models are trained using the VBL loss. For the simplified objective, DAED trained with the same number of diffusion steps yields slightly lower performance than standard DDGMs. As indicated by the Precision/Recall, generations from DAED are as precise as those from DDGM. However, they lack certain diversity, probably due to the smoothing effect of the DAE part. Detailed results for other setups are presented in Appendix 5.9.1.[3]

[2] Generations for all datasets are presented in Appendix 5.9.3

[3] In Appendix 5.9.7 we show that increasing the number of parameters of DDGMs to be comparable to DAED does not lead to significant performance improvements.

5.7.3. Does the noise removal in DDGMs generalize to other data distributions?

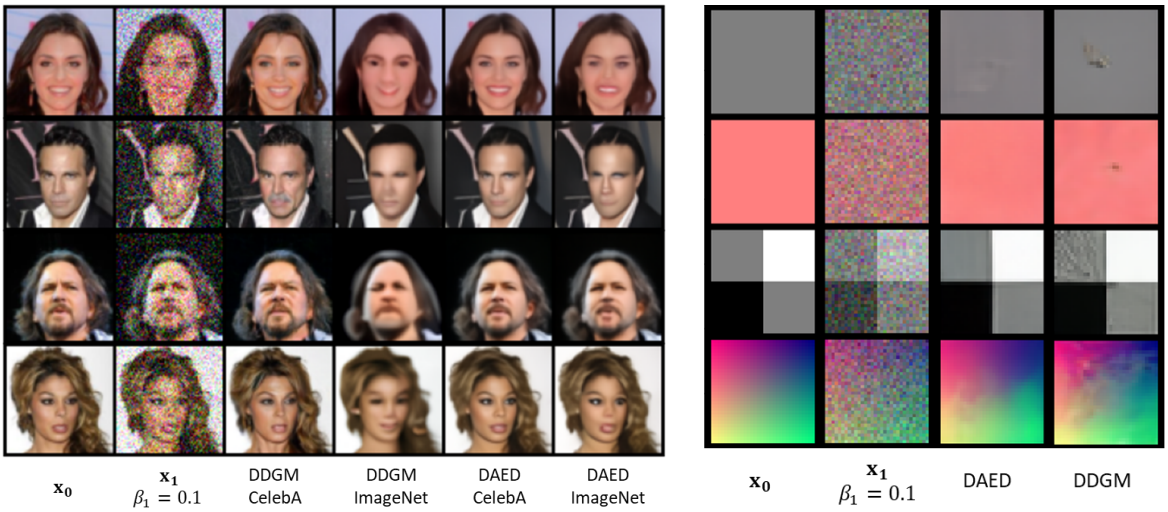
The last question we are interested in is the generalizability of DDGMs to other data distributions. We refer to this concept as *transferability* for short. In other words, the goal of this experiment is to determine whether we can reuse a model or its part on new data with as good performance as possible. In this experiment, we rely on the results presented in Section 5.7.1 where roughly the first 10% of steps could be seen as the denoising part. To further strengthen this perspective, we also utilize DAED with an explicit division into the denoising and generating parts.

Table 5.7.2. Reconstruction errors measured by MAE (↓), MS-SSIM (↑) for images noised with $\beta_1 = 0.1$. *To evaluate models trained on CIFAR10, we downscale CelebA to 32×32. **Best results in bold.**

	Target dataset	CIFAR10		CIFAR100		CelebA*	
Source Dataset	Model	MAE	MS-SSIM	MAE	MS-SSIM	MAE	MS-SSIM
CIFAR10	DDGM VLB	0.091	0.94	0.097	0.94	0.093	0.95
	DDGM Simple	0.085	0.95	0.097	0.94	0.096	0.95
	DAED	**0.065**	**0.97**	**0.074**	**0.97**	**0.068**	**0.97**
ImageNet	DDGM VLB	0.113	0.93	0.110	0.93	0.077	0.96
	DDGM Simple	0.113	0.94	0.111	0.93	0.068	0.96
	DAED	**0.071**	**0.97**	**0.071**	**0.97**	**0.050**	**0.98**

First, we consider the case in which we compare the reconstruction errors measured by the MAE and the MS-SSIM. In this scenario, we train a DDGM on a source dataset and then assess it on a target dataset. We use CIFAR10 or ImageNet (32x32 or 64x64) as source data and CIFAR10, CIFAR100, or CelebA as target data. For each image from the target dataset, we apply the DAE part of DAED to obtain the reconstruction or 793 steps of the forward and backward diffusion in the case of the DDGM, which corresponds to the same level of added noise. For this experiment, we use the pre-trained DDGM from Nichol and Dhariwal (2021) that consists of 4000 steps and uses the cosine noise scheduler. The results are outlined in Table 5.7.2. First of all, there is no significant difference in the performance of DDGMs trained with either the VBL objective or the simplified objective. They achieve a quite satisfactory MAE and MS-SSIM scores. However, DAED outperforms the DDGMs, obtaining much better transferability. We explain it by the fact that probably, with each step in the denoising part DDGM adds details that are typical for source data while DAED focuses on removing noise and produces a smoother output. This outcome may further suggest that splitting DDGMs into two parts with two separate parameterizations is reasonable and even beneficial.

To get further insight into the transferability behavior, we present a few (non-cherry–picked) examples from CelebA in Figure 5.7.4.a and four toy examples in Figure 5.7.4.b. We use the same setup as explained in the previous paragraph (i.e., the pre-trained DDGM provided by Nichol and Dhariwal (2021)), and the images are noised with $\beta_1 = 0.1$. In columns 3–6 in Figure 5.7.4.a, we present reconstructions for the DDGM trained on CelebA, the DDGM trained on ImageNet, DAED trained on CelebA, and DAED trained on ImageNet, respectively. It becomes apparent that the DDGM trained on CelebA denoises the image by generating new details while DAED denoises by smoothing. Interestingly, DAED performs better than the DDGM when we use ImageNet-trained models to denoise CelebA. In Figure 5.7.4.b, we depict several toy examples that were denoised with the DDGM and DAED trained on CIFAR10. We see that the DDGM adds many details that are artifacts from the source data. It seems that DAED does not suffer from that behavior.



(a) Reconstructions on CelebA (b) Toy examples

Figure 5.7.4. (a) Denoising of image with 0.1 noise using either DAED or the corresponding number of the DDGM steps. (b) Four noisy toy examples denoised by DAED and the DDGM.

5.8. Conclusion

In this work, we investigate the generative and denoising capabilities of the Diffusion-based Deep Generative Models. We observe and experimentally validate that it is reasonable to understand DDGMs as a combination of two parts. The first one generates noisy samples from the pure noise by inputting more signal from a learned data distribution, while the second one removes the remaining noise from the signal. Although for standard DDGMs, the exact switching point between those two parts is fluid, we propose a new approach dubbed DAED that is explicitly built as a combination of a generative component (a DDGM) and a denoising one (a DAE).

In the experiments, we observe that DAED simplifies training with a standard VLB loss function that leads to improved performance. On the other hand, with increasing noise processed by DAE, DAED smoothens the generations resulting in lower performance when training with the simplified objective. We further show that DDGMs, and DAED especially, generalize well to unseen data, what opens new possibilities for further research in terms of transfer or continual learning of DDGMs.

5.9. Appendix

5.9.1. Additional experiments

In this section, we present extended evaluation of all models introduced in the main work. Following Nichol and Dhariwal (2021), we show the assessment of generations quality in terms of additional metrics namely Inception Score (Salimans et al., 2016) and spatial Fréchet Inception Distance (Nash et al., 2021) – a version of standard FID score but based on spatial image features.

Table 5.9.1. Extended evaluation results for CIFAR10 dataset.

Model			CIFAR-10				
	Loss	T	IS ↑	FID ↓	sFID ↓	Prec ↑	Rec ↑
DDGM	VLB	1000	7.6	26.1	10.5	54	55
DAED $\beta_1 = 0.1$	VLB	900	**8.2**	20.4	16.1	59	46
DAED $\beta_1 = 0.025$	VLB	979	7.7	22.4	15.8	57	53
DAED linear	VLB	999	8.1	**14.5**	**9.8**	**60**	**59**
DDGM	Simple	1000	**9.5**	**7.2**	**8.6**	65	**61**
DAED $\beta_1 = 0.2$	Simple	891	7.8	29.4	24.7	53	40
DAED $\beta_1 = 0.1$	Simple	900	8.0	19.0	14.9	62	50
DAED $\beta_1 = 0.025$	Simple	979	8.6	14.2	14.6	60	53
DAED $\beta_1 = 0.001$	Simple	999	9.1	14.9	10.1	**66**	54

Table 5.9.2. Extended evaluation results for CelebA dataset. Additionally to standard models, we also include evaluation for DAED setup where DAE model is trained only on ImageNet dataset.

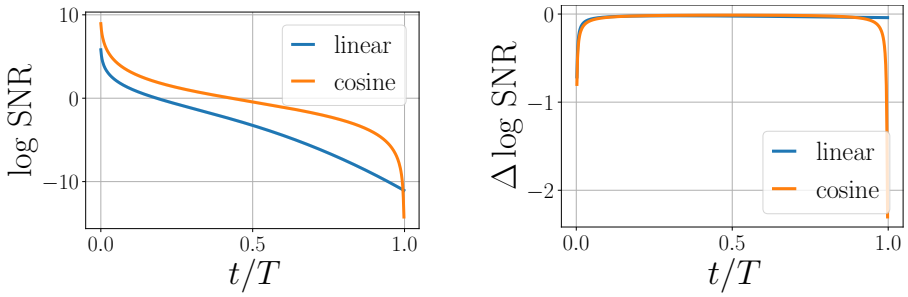
Model			CelebA				
	Loss	T	IS ↑	FID ↓	sFID ↓	Prec ↑	Rec ↑
DDGM	VLB	1000	2.4	23.1	37.3	51	21
DAED $\beta_1 = 0.1$	VLB	900	**2.9**	18.2	23.9	63	**31**
DAED $\beta_1 = 0.025$	VLB	979	2.7	25.4	35.8	64	17
DAED linear	VLB	1000	2.6	**16.8**	**23.6**	**70**	27
DDGM	Simple	1000	**3.0**	**6.1**	14.7	66	**56**
DAED $\beta_1 = 0.2$	Simple	890	2.7	21.0	31.2	63	22
DAED $\beta_1 = 0.1$	Simple	900	**3.0**	17.0	23.3	66	31
DAED $\beta_1 = 0.025$	Simple	979	2.7	15.1	17.6	64	38
DAED $\beta_1 = 0.001$	Simple	999	2.8	6.2	**11.0**	**69**	55
DAED (IN) $\beta_1 = 0.1$	Simple	900	2.9	25.6	30.5	44	29

Table 5.9.3. Extended evaluation results for Fashion MNIST dataset.

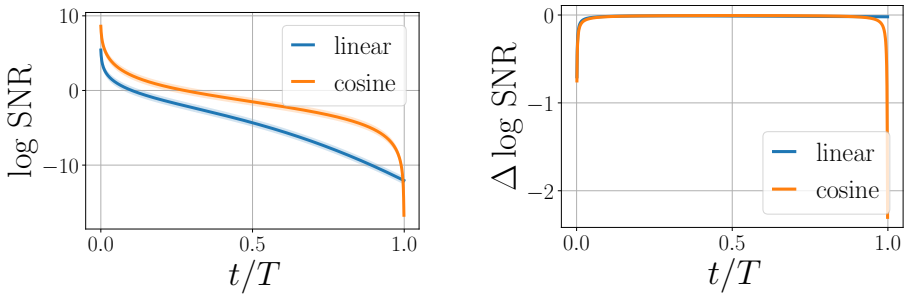
			Fashion Mnist				
	Loss	T	IS ↑	FID ↓	sFID ↓	Prec ↑	Rec ↑
DDGM	vlb	500	**4.1**	8.9	**11**	68	53
DAED $\beta_1 = 0.1$	vlb	468	4.06	9.1	13	**71**	60
DAED $\beta_1 = 0.025$	vlb	489	4.02	9.7	**11**	70	62
DAED linear	vlb	499	**4.1**	**7.5**	11.3	70.5	**64**
DDGM	Simple	500	**4.3**	7.8	**9.03**	71.5	**65.3**
DAED $\beta_1 = 0.3$	Simple	426	3.78	18	24	73.8	41
DAED $\beta_1 = 0.2$	Simple	445	3.87	14	20	**74.8**	47
DAED $\beta_1 = 0.1$	Simple	468	3.95	9.6	11.2	73.2	58.4
DAED $\beta_1 = 0.025$	Simple	489	4.05	7.36	13	73	61
DAED $\beta_1 = 0.001$	Simple	499	**4.3**	**5.7**	11.3	69.3	64.2

5.9.2. Signal-to-noise ratio detailed plots

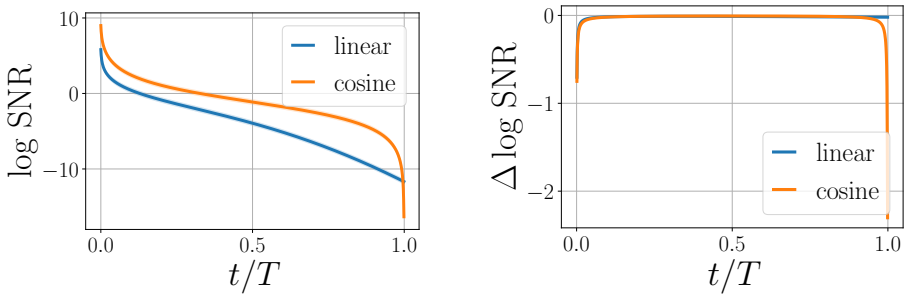
In this section we present detailed signal-to-noise ratio (SNR) plots that are used for analysis in Sec. 5.5 for all evaluated datasets. Independently on the original dataset, SNR changes in the similar manner – with the most drastic loss in the first 10% steps.



(a) FashionMNIST



(b) CIFAR10



(c) CelebA

Figure 5.9.1. Signal-to-noise ratio and its discrete derivative for each of the three datasets: (a) FashionMNIST, (b) CIFAR10 and (c) CelebA.

5.9.3. Examples of generations

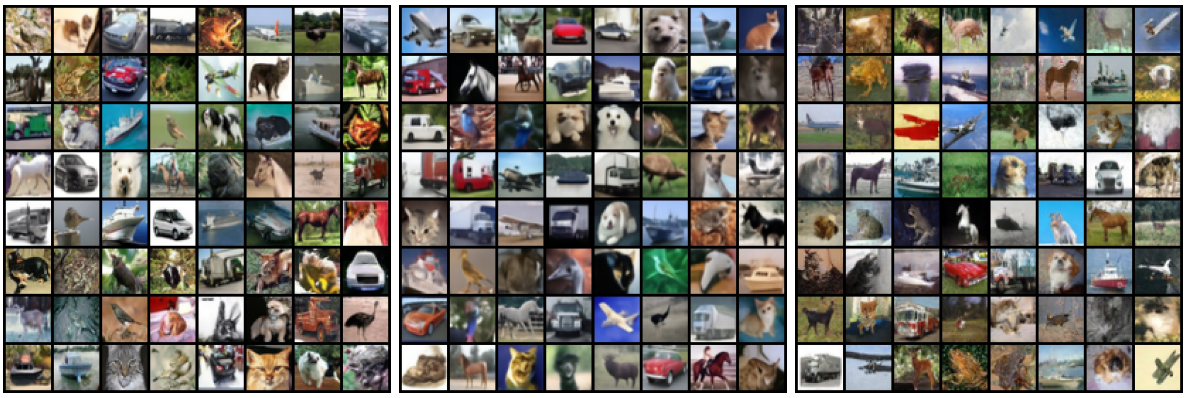
In this section we present generations for all datasets with different models we compare in this work.

5.9.4. Training Dynamics

How does the objective of a diffusion model change in time? In the standard DDGM setup, a single model is optimized with a joint loss from all of the diffusion steps. However, as depicted in Fig 5.9.8a, different parts of the diffusion contribute to the sum differently. In fact, the first step of the diffusion is already responsible for 75% of the whole training loss, while first 1% of steps contributes to over the 90% of the training objective. This observation implies that a single neural network applied to all diffusion steps is mostly optimized to denoise the initial steps. In Fig. 5.9.7

(a) DDGM **(b)** DAED $\beta_1 = 0.1$ **(c)** DAED $\beta_1 = 0.001$

Figure 5.9.2. Generations from different models trained on FashionMNIST dataset. All models were trained with Simple loss function.



(a) DDGM **(b)** DAED $\beta_1 = 0.1$ **(c)** DAED $\beta_1 = 0.001$

Figure 5.9.3. Generations from different models trained on CIFAR10 dataset. All models were trained with Simple loss function.

(a) DDGM **(b)** DAED $\beta_1 = 0.1$ **(c)** DAED $\beta_1 = 0.001$

Figure 5.9.4. Generations from different models trained on CelebA dataset. All models were trained with Simple loss function.

(a) DDGM **(b)** DAED $\beta_1 = 0.1$ **(c)** DAED $\beta_1 = 0.001$

Figure 5.9.5. Generations from different models trained on CelebA dataset with original VLB loss function.

(a) DAED $\beta_1 = 0.1$

Figure 5.9.6. Generations from DAED model where DDGM part was trained on CelebA dataset while DAE on ImageNet.

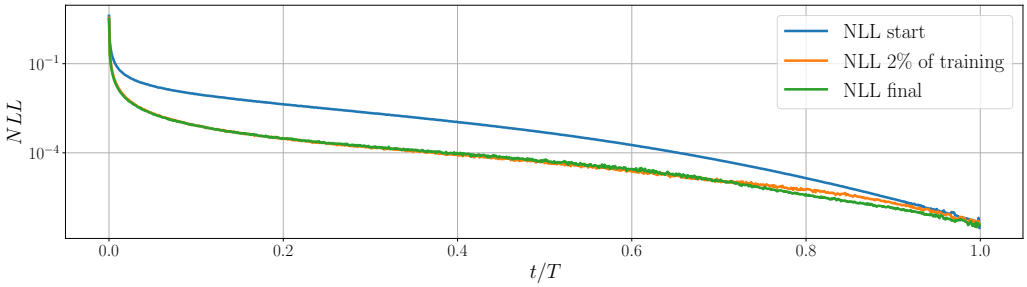
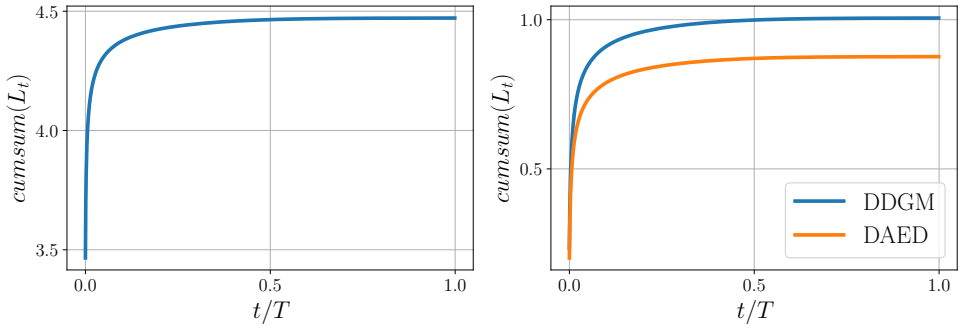


Figure 5.9.7. Dynamics of the negative log likelihood for different steps of standard DDGM trained on CIFAR10 with VLB objective. Already after 2% of training time, p_θ converges to very low loss values (below 0.001) for all of the training steps above 0.1T.



(a) NLL Cumsum **(b)** NLL Cumsum without t_1

Figure 5.9.8. The cumulative sum of the negative log likelihood for different steps of a standard DDGM trained on CIFAR10 with the VLB objective (left), and the same cumulative sum without the first diffusion step in comparison to DAED with exactly the same β scheduler.

we present how this loss contribution changes over time. Surprisingly, only 2% of the training time is needed to align latter 90% of training steps to the loss value below 0.01. These observations led to the emergence of cosine scheduler (Nichol and Dhariwal, 2021) where authors change the noise scheduler to increase the number of steps with higher loss values.

In this work, we propose to tackle this problem from a different perspective and to analyze what happens if we detach the loss from initial diffusion steps from the total sum. In Figure 5.9.8b, we compare how such a detachment of the first step of 1000-stepped DDGM with DAED influence the loss value on the remaining 999 steps. As depicted in DAED, the loss converges to lower values that explains the improvement of the performance of DAED when training with the VLB loss.

5.9.5. Training Hyperparameters

In all of our experiments, we follow Nichol and Dhariwal (2021). We train all models with U-Net architecture, with three or four depth levels (depending on a

Table 5.9.4. DDGM and DAED hyperparameters for different datasets

Dataset	train-steps	depth	channels
FashionMNIST	100k	3	64, 128, 128
CIFAR10	500k	3	128, 256, 256, 256
CelebA	200k	4	128, 256, 384, 512

dataset), with three residual blocks each, with a given number of filters depending on the dataset – as presented in 5.9.4. In all of our models, we use time embeddings and attention-based layers with three attention heads in each model.

We optimize our models on the basis of randomly selected diffusion steps. For the standard DDGM, for simplicity, we use a uniform sampler, while for DAED, we propose a weighted uniform sampler, where the probability of sampling from a given step t is proportional to the given β_t. This also applies to the Denoising Autoencoder as a part of DAED that is updated accordingly to the new sampler. We update models parameters with AdamW (Loshchilov and Hutter, 2017) optimizer for a given number of batches as presented in 5.9.4. To prevent our model from overfitting, we use dropout (Hinton et al., 2012) with probability $p = 0.3$. Detailed implementation choices, examples of training runs and models can be found in the attached code repository.

5.9.6. Computational details

Diffusion-based deep generative models are known for being computationally expensive. For our training, we used Nvidia Titan RTX GPUs for complex datasets (CIFAR, CelebA, ImageNet) and Nvidia GeForce 1080Ti for FashionMNIST. Full training of our model on FashionMNIST for 100k steps on a single GPU took approximately 35 hours. For CIFAR and CelebA we used parallel computation based with four GPUs. Full training with this setup took approximately 48 hours. Those estimates are valid for training of both DDGM and DAED.

5.9.7. A comparison between DAED and DDGMs with more parameters

The DAED model uses two separate UNet models for the generative and denoising parts. As a result, it has twice as many parameters as a DDGM. In Table 5.9.5 we compare DAED with DDGMs that have a comparable number of parameters. We double the size of the UNet model for vanilla DDGM in two setups. In the first one we increase the number of convolution channels, while in the second one, we double the number of residual blocks.

Table 5.9.5. A comparison of DAED with DDGMs of different sizes on the FashionMNIST dataset.

	Total Params (mln.)	Inference Time (sec. per sample)	FID ↓	Prec ↑	Rec ↑
DDGM	8.8	0.65	7.8	72	65
DDGM 1.5× channels	19.8	0.84	8	74	65
DDGM 2× blocks	15.1	1.19	7.5	66	66
DAED	17.6	0.66	5.7	69	64

The results in Table 5.9.5 suggest that the performance of DAED over DDGMs cannot be attributed purely to the larger number of parameters. As we increase the number of layers of the UNet used by the DDGM, we see only a slight improvement of the performance. Furthermore, a larger UNet leads to a significant increase in the inference time compared to the smaller DDGM and DAED.

6. Learning Data Representations with Joint Diffusion Models

Title	Learning Data Representations with Joint Diffusion Models
Authors	Kamil Deja, Tomasz Trzciński, Jakub M. Tomczak
Conference	*accepted for:* European Conference on Machine Learning
Year	2023

Preface

In the previous chapter, we analysed how the behaviour of Diffusion-Based Deep Generative models changes with diffusion timesteps. We showed that the denoising decoder first learns meaningful data representations and generates new data features from random noise, while the latest steps are used to remove the remaining noise artefacts in a purely deterministic way. In the following work, we further analyse this process and look at the data representations in the DDGMs from a different perspective – as the internal activations within the denoising decoder. We first focus on their structure, analysing how the encoded latent features change over time, and then we show how we can use the same features outside of the generative task.

To that end, we first postulate that in a UNet model, most commonly used as a decoder of a DDGM, we can identify the latent data representations encoded by the first part of the model known as Encoder. Such an approach resembles the way how a generative autoencoder creates latent representations. Following this observation, in the first analysis, we show that latent representations encode meaningful data features. For example, they can be successfully used for classification even when a denoiser is trained without supervision. Then, we analyse how those representations change with time. Similarly to the previous work, we observe a non-stationary behaviour of the decoder. In particular, it encodes low-grain data features in the early steps of the backward diffusion process, while high-frequency details are represented later. Therefore, we show that not only do diffusion models learn meaningful data representations, but they also disentangle them in the process.

On top of our analysis, we introduce a joint diffusion model where we propose to use the extracted representations as an input for a classifier and optimise the whole network jointly. This way, we use one parameterisation to model both the probability of an example $p(\mathbf{x})$ and a marginal probability of its class assignment $p(\mathbf{y}|\mathbf{x})$. We present several use cases where we can benefit from this formulation and use the shared data representations in practical scenarios such as semi-supervised learning, unsupervised domain adaptation or counterfactual examples generation.

Abstract

We introduce a joint diffusion model that simultaneously learns meaningful internal representations fit for both generative and predictive tasks. Joint machine learning models that allow synthesising and classifying data often offer uneven performance between those tasks or are unstable to train. In this work, we depart from a set of empirical observations that indicate the usefulness of internal representations built by contemporary deep diffusion-based generative models in both generative and predictive settings. We then introduce an extension of the vanilla diffusion model with a classifier that allows for stable joint training with shared parameterisation between those objectives. The resulting joint diffusion model offers superior performance across various tasks, including generative modelling, semi-supervised classification, and domain adaptation.

6.1. Introduction

Training a single machine learning model that can jointly synthesise new data as well as to make predictions about input samples remains a long-standing goal of machine learning (Jebara, 2012; Lasserre et al., 2006). Shared representations created with a combination of those two objectives promise benefits on many downstream problems such as calibration of model uncertainty (Chapelle et al., 2009), semi-supervised learning (Kingma et al., 2014), unsupervised domain adaptation (Ilse et al., 2020) or continual learning (Masarczyk et al., 2021).

Therefore, since the introduction of deep generative models such as Variational Autoencoders (VAEs) (Kingma and Welling, 2014), a growing body of work takes advantage of shared deep neural network-based parameterisation and latent variables to build joint models. For instance, (Ilse et al., 2020; Tulyakov et al., 2017; Knop et al., 2020; Yang et al., 2022a) stack a classifier on top of latent variables sampled from a shared encoder. Similarly, Nalisnick et al. (2019b) and Perugachi-Diaz et al. (2021) use normalising flows to obtain an invertible representation that is further fed to a classifier. However, these approaches require modifying the log-likelihood function by scaling either the conditional log-likelihood or the marginal log--likelihood. This idea, known as hybrid modelling (Lasserre et al., 2006), leads to the situation where models concentrate either on synthesising data or predicting but not on both of those tasks simultaneously.

We address existing joint models' limitations and leverage the recently introduced diffusion-based deep generative models (DDGM) (Sohl-Dickstein et al., 2015; Dhariwal and Nichol, 2021; Kingma et al., 2021). This new family of methods has

become popular because of the unprecedented quality of the samples they generate. However, relatively little attention was paid to their inner workings, especially to the internal representations built by the DDGMs. In this work, we fill this gap and empirically analyse those representations, validating their usefulness for predictive tasks and beyond. Then, we introduce a joint diffusion model, where a classifier shares the parameterisation with the UNet encoder by operating on the extracted latent features. This results in meaningful data representations shared across discriminative and generative objectives.

We validate our approach in several use cases where we show how one part of our model can benefit from the other. First, we investigate how DDGMs benefit from the additional classifier to conditionally generate new samples or alter original images. Next, we show the performance improvement our method brings in the classification task. Finally, we extend the evaluation of our joint diffusion model to semi-supervised learning, domain adaptation, and counterfactual explanations. For all of those tasks, our method does not require any problem-specific adjustments, which confirms the flexibility of our approach.

We can summarise the contributions of our work as follows:

- We provide empirical observations with insight into representations built internally by diffusion models, on top of which we introduce a joint classifier and diffusion model with shared parameterisation.
- We introduce a conditional sampling algorithm where we optimise internal diffusion representations with a classifier.
- We prove that our solution work with several use cases including the semi--supervised learning, domain adaption and counterfactual explanations.

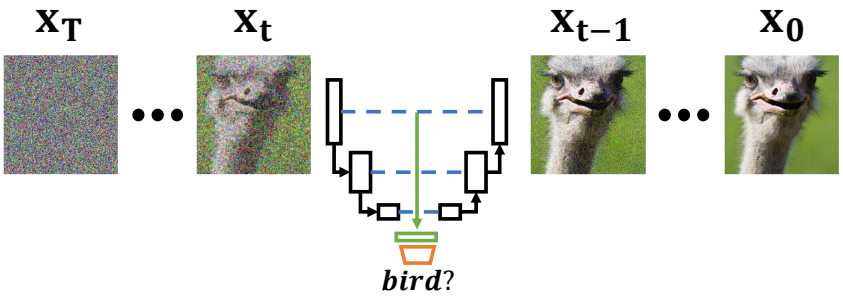


Figure 6.1.1. The overview of our method. We propose to jointly train the diffusion model and the classifier using a single parameterisation with a shared UNet architecture.

6.2. Background

Joint models Let us consider two random variables: $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$. For instance, in the classification problem we can have $\mathcal{X} = \mathbb{R}^D$ and $\mathcal{Y} = \{0, 1, \ldots, K-1\}$. The joint distribution over these random variables could be factorised in one of the following two manners:

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y) \tag{6.1}$$

$$= p(y|\mathbf{x})\ p(\mathbf{x}). \tag{6.2}$$

In Eq. (6.2), we get the conditional distribution $p(y|\mathbf{x})$ (e.g., a classifier) and the marginal distribution $p(\mathbf{x})$. For prediction, it is enough to learn the conditional distribution, which is typically parameterised with neural networks. However, training the joint model with shared parameterisation has many advantages since one part of the model can positively influence the other.

6.3. Related Work

Diffusion models There are several extensions to the baseline DDGM setup that aim to improve the quality of sampled generations (Ho et al., 2020; Huang et al., 2021; Kingma et al., 2021; Song and Ermon, 2019; Song et al., 2020b). With extensions to the classifier and classifier-free guidance as described in Chapter 3.2. Among those works, Dhariwal and Nichol (2021) introduce a classifier-guided generation, where a gradient from an externally and independently trained classifier is added in the process of backward diffusion to guide the generation towards a target class. On top of this approach, Augustin et al. (2022) present a tool for investigating the decision of a classifier by generating visual counterfactual explanations with a diffusion mode. In this work, we simplify both of those methods benefiting from training a joint model with representations shared between a diffusion model and a classifier.

Diffusion models and UNet representations Some works tackle the problem of data representation with diffusion models. We describe them closely in Chapter 3.2, while in this work, we show that indeed diffusion models learn useful representations. We further take advantage of that in utilising a shared parameterisation between a diffusion model and a classifier in a joint model.

Joint training Apart from latent variable joint models, Grathwohl et al. (2019b) show that it is possible to use a shared parameterisation (a neural network-based classifier) to formulate an energy-based model. This Joint Energy-based Model

(JEM) could be seen as a classifier if a softmax function is applied to logits or a generator if a Markov-chain Monte Carlo method is used to sample from the model. Although it obtains strong empirical results, gradient estimators used to train JEM are unstable and prone to diverging when optimisation parameters are not perfectly tuned, which limits the robustness and applicability of this method. Alternatively, Introspective Neural Networks could be used for generative modelling and classification by applying a single parameterisation (Jin et al., 2017; Lazarow et al., 2017; Lee et al., 2018). The idea behind this class of models relies on utilising a training procedure that combines adversarial learning and contrastive learning. Similarly to JEMs, sampling is carried out by running an MCMC method. Grathwohl et al. (2021) improve the performance of JEM by introducing a variational-based approximator (VERA) instead of MCMC. Similarly, Yang and Ji (2021) introduce JEM++, an improvement over the JEM's generative performance by applying a proximal SGLD-based generation, and classification accuracy with informative initialisation. From a conceptually different perspective, Yang et al. (2022b) propose an implementation of a joint model based on the Vision Transformer (Dosovitskiy et al., 2020) architecture, that yields state-of-the-art result in terms of image classification. Here, we propose to combine standard diffusion models with classifiers by sharing their parameterisation. Thus, our training is entirely based on the log-likelihood function and end-to-end, while sampling is carried out by backward diffusion instead of any MCMC algorithm.

6.4. Diffusion Models Learn Data Representations

As mentioned earlier, learning useful data representations is important for having a good generator or classifier. Ideally, we would like to train a joint model that allows us to obtain proper representations for both $p(y|\mathbf{x})$ and $p(\mathbf{x})$ simultaneously. In this work, we investigate parameterisations of DDGMs and, in particular, the use of an autoencoder as a denoising decoder $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. Within this architecture, the denoising function can be decomposed into two parts: encoding of the image at the current timestep into a set of features $\mathcal{Z}_t = e(\mathbf{x}_t)$ and then decoding it to obtain $\mathbf{x}_{t-1} = d(\mathcal{Z}_t)$. In particular, for the UNet architecture, a set of features obtained from an input is a structure composed of several tensors with image features encoded to different levels, $\mathcal{Z}_t = \{\mathbf{z}_t^1, \mathbf{z}_t^2 \dots \mathbf{z}_t^n\}$. For simplification, for all further experiments, we propose to pool features encoded by the same filter and concatenate the averaged representations into a single vector $\mathbf{z}_t$, as presented in Fig. 6.4.1 for $n = 3$. In particular, we can use average pooling to select average convolutional filter activations to the whole input. Details of this procedure are described in Appendix 6.8.1.

6.4.1. UNet representations are useful for prediction

First, we would like to verify whether averaged representations $\mathbf{z}_0$ extracted from an original image $\mathbf{x}_0$ by the UNet contain information that is in some sense predictive. For that, we measure the performance of an MLP-based classifier fed with $\mathbf{z}_0$.

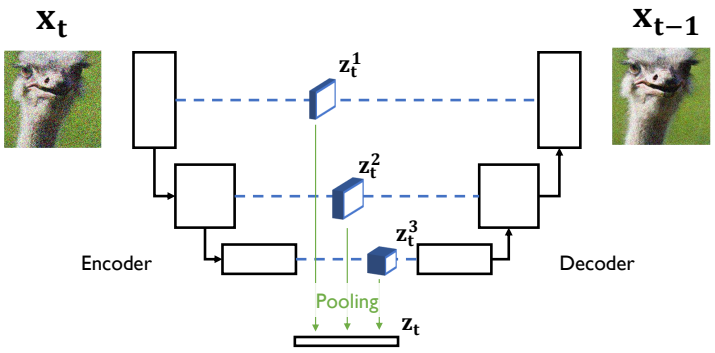


Figure 6.4.1. Data representation z_t in a UNet-based diffusion model.

As presented in Fig 6.4.2a, representations encoded in $\mathbf{z}_0$ are indeed very informative and, in some cases (e.g., CIFAR-10), could lead to performance comparable to a stand-alone classifier with the same architecture as the combination of the UNet encoder and MLP but trained with the standard cross-entropy loss function. This observation is in line with Baranchuk et al. (2021), where the same activations from the pre-trained diffusion model were used for semantic image segmentation.

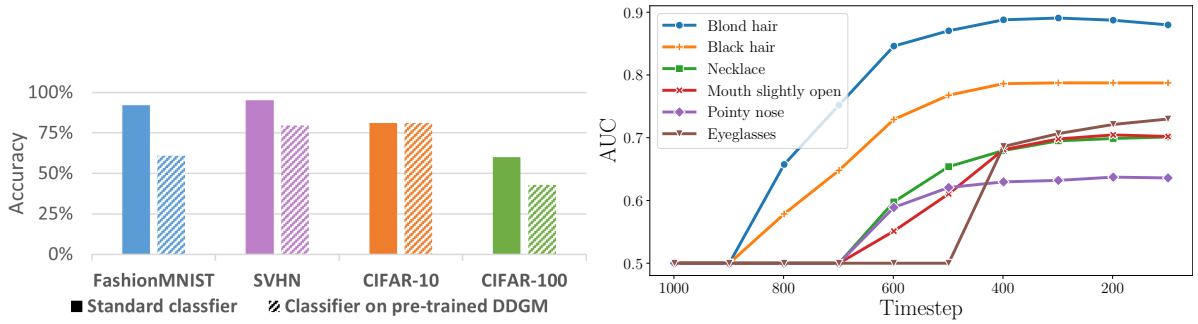


Figure 6.4.2. (a) The test-set accuracy of a stand-alone classifier compared to a classifier trained on top of data representations from a pre-trained diffusion model extracted from original images $\mathbf{x}_0$. (b) The area under the ROC curve (AUC) for logistic regression models fit on data representations extracted with a pre-trained diffusion model at ten different diffusion timesteps. High-grained features are already distinguishable at late diffusion steps (closer to random noise), while low-grained features are only represented at the earlier stage of the forward diffusion.

6.4.2. Diffusion models learn features of increasing granularity

The next question is how the data representations $\mathbf{z}_t$ differ with diffusion timesteps t. To investigate this issue, we train an unsupervised DDGM on the CelebA dataset, which we then use to extract the features $\mathbf{z}_t$ at different timesteps. On top of those representations, we fit a binary logistic regression classifier for each of the 40 attributes in the dataset. In Fig. 6.4.2b, we show the performance of those regression

models for 6 different attributes when calculated on top of representations from ten different diffusion timesteps. We observe that the model learns different data features depending on the amount of noise added to the original data sample. As presented in Fig. 6.4.2b, high-grained data features such as hair colour start to emerge at late diffusion steps (closer to the noise), while low-grained features (e.g., necklace or glasses) are not present until the early steps. This observation is in line with the works on denoising autoencoders where authors observe similar behaviour for denoising with different amounts of added noise Chandra and Sharma (2014); Geras and Sutton (2014); Zhang and Zhang (2018).

6.5. Method

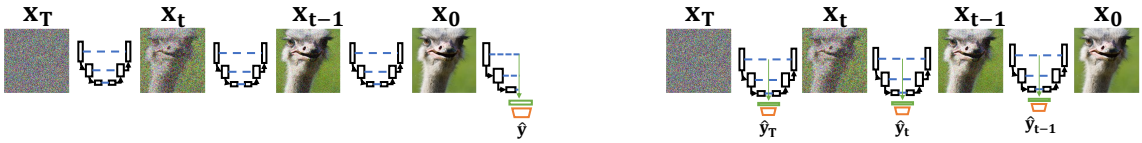
Taking into account the observations described in Section 6.4, we propose to train a joint model that is composed of a classifier and a DDGM. Specifically, we propose to use a shared parameterisation, namely, a shared encoder of the UNet architecture that serves as the generative part and for calculating pooled features for the classifier.

6.5.1. Joint Diffusion Models: DDGMs with classifiers

Following the procedure introduced in Sec. 6.4, we pool the latent representations of the data from different levels of the UNet architecture into one vector $\mathbf{z}$. On top of this vector, we build a classifier model trained to assign a label to the data example represented by the vector $\mathbf{z}$.

In particular, we consider the following parameterisation of a denoising diffusion model within a single diffusion timestep t, $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$. We distinguish the encoder e_ν with parameters ν that maps input $\mathbf{x}_t$ into a set of vectors $\mathcal{Z}_t = e_\nu(\mathbf{x}_t)$, where $\mathcal{Z}_t = \{\mathbf{z}_t^1, \mathbf{z}_t^2 \dots \mathbf{z}_t^n\}$, i.e., a set of representation vectors derived from each depth level of the UNet architecture. The second component of the denoising diffusion model is the decoder d_ψ with parameters ψ that reconstructs feature vectors into a denoised sample, $\mathbf{x}_{t-1} = d_\psi(\mathcal{Z}_t)$. Together the encoder and the decoder form the denoising model p_θ with parameters $\theta = \{\nu, \psi\}$. Next, we introduce a third part of our model, which is the classifier g_ω with parameters ω that predicts target class $\hat{y} = g_\omega(\mathcal{Z}_t)$. The first layer of the classifier is the average pooling that results in a single representation $\mathbf{z}_t$.

In our approach, we consider a classifier that takes the original image $\mathbf{x}_0$ for which a vector of probabilities is returned φ and eventually the final prediction is calculated, $\hat{y} = g_\omega(\mathbf{x}_0)$. The visualisation of our shared parameterisation is presented



(a) The parameterisation of our joint diffusion *(b)* Additional noisy classifiers

Figure 6.5.1. The parameterisation of our joint diffusion model. *(a)* Each step in the backward diffusion is parameterised by a shared UNet. The classifier uses the encoder of the UNet together with the average pooling (green) and additional layers (yellow). *(b)* An alternative training that additionally uses the classifier for noisy images $\mathbf{x}_t$ ($t > 0$).

in Figure 6.5.1*(a)*. As a result, our model could be written as follows $p_{\nu,\psi,\omega}(\mathbf{x}_{0:T}, y) = p_{\nu,\omega}(y|\mathbf{x}_0)\ p_{\nu,\psi}(\mathbf{x}_{0:T})$, and applying the logarithm yields:

$$\ln p_{\nu,\psi,\omega}(\mathbf{x}_{0:T}, y) = \ln p_{\nu,\omega}(y|\mathbf{x}_0) + \ln p_{\nu,\psi}(\mathbf{x}_{0:T}). \tag{6.3}$$

The logarithm of the joint distribution (6.3) could serve as the training objective in which $\ln p_\theta(\mathbf{x}_{0:T})$ could be either approximated by the ELBO for the diffusion-based model in (2.9) or the simplified objective with (2.13)). In this paper, we follow the simplified objective:

$$L_{t,\text{diff}}(\nu,\psi) = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}\|^2\right], \tag{6.4}$$

where $\hat{\epsilon}$ is a prediction from the decoder:

$$\{\mathbf{z}_t^1, \mathbf{z}_t^2 \dots \mathbf{z}_t^n\} = e_\nu\left(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t\right) \tag{6.5}$$

$$\hat{\boldsymbol{\epsilon}} = d_\psi(\{\mathbf{z}_t^1, \mathbf{z}_t^2 \dots \mathbf{z}_t^n\}). \tag{6.6}$$

For the classifier, we use the logarithm of the categorical distribution, i.e., the crossentropy loss:

$$L_{\text{class}}(\nu,\omega) = -\mathbb{E}_{\mathbf{x}_0,y}\left[\sum_{k=0}^{K-1} \mathbb{1}[y=k] \log \frac{\exp(\varphi_k)}{\sum_{c=0}^{K-1}\exp(\varphi_c)}\right], \tag{6.7}$$

where y is a target class, φ is a vector of probabilities returned by the classifier $g_\omega(e_\nu(\mathbf{x}_0))$, and $\mathbb{1}[y=k]$ is the indicator function that is 1 if y equals k, and 0 otherwise.

The final loss function in our approach is then the following:

$$\begin{aligned} L(\nu,\psi,\omega) = L_{\text{class}}(\nu,\omega) + \\ - L_0(\nu,\psi) - \sum_{t=2}^{T} L_{t,\text{diff}}(\nu,\psi) - L_T(\nu,\psi). \end{aligned} \tag{6.8}$$

We optimize the objective in (6.8) jointly with a single optimizer over parameters $\{\nu, \psi, \omega\}$.

6.5.2. An alternative training of joint diffusion models

The training of the proposed approach over a batch of data is straightforward. For a sampled pair $(\mathbf{x}_0, y)$, the example $\mathbf{x}_0$ is first noised with a forward diffusion to a random timestep, $\mathbf{x}_t$ so that the training loss for the denoising model is a Monte-Carlo approximation of the sum over all timesteps. Then $\mathbf{x}_0$ is fed to a classifier that returns probabilities φ, and the cross-entropy loss is calculated for given y.

However, as discussed in Section 6.4.2, the diffusion model trained even in a fully unsupervised manner provides data representations related to the different granularity of input features at various diffusion timesteps. Considering this, we can improve the robustness of our method by applying the same classifier to intermediate noisy images $\mathbf{x}_t$ $(0 < t < T)$, which by reason adds the cross-entropy losses for $\mathbf{x}_t$, namely:

$$L^t_{\text{class}}(\nu, \omega) = -\mathbb{E}_{\mathbf{x}_0, y}\left[\sum_{k=0}^{K-1} \mathbb{1}[y = k] \log \frac{\exp\left(\varphi^t_k\right)}{\sum_{c=0}^{K-1} \exp\left(\varphi^t_c\right)}\right], \tag{6.9}$$

where φ^t_k is a vector of probabilities given by $g_\omega(e_\nu(\mathbf{x}_t))$. Then the extended objective (6.8) is the following:

$$L_{\mathcal{T}}(\nu, \psi, \omega) = L(\nu, \psi, \omega) + \sum_{t \in \mathcal{T}} L^t_{\text{class}}(\nu, \omega), \tag{6.10}$$

where $\mathcal{T} \subseteq \{1, 2, \ldots, T\}$ is the set of timesteps. These additional *noisy classifiers* are schematically depicted in Figure 6.5.1*(b)* in which we highlight that the model is reused across various noisy images. It is important to mention that the noisy classifiers serve only for training purposes; they are not used for prediction. This procedure is similar to the data augmentation technique, where random noise is added to the input Sietsma and Dow (1991).

6.5.3. Conditional sampling in joint diffusion models

To improve the quality of samples generated by DDGM, Dhariwal and Nichol (2021) propose a classifier guidance approach, where an externally trained classifier can be used to guide the generation of the DDGM trained in an unsupervised way towards the desired class. In the standard DDGM, at each backward diffusion step,

an image is sampled from the output of the diffusion model p_θ according to the following formula:

$$\begin{aligned} \mu, \Sigma &\leftarrow \mu_\theta(\mathbf{x}_t), \Sigma_\theta(\mathbf{x}_t) \\ \mathbf{x}_{t-1} &\leftarrow \text{sample from } \mathcal{N}(\mu, \Sigma) \end{aligned} \tag{6.11}$$

Dhariwal and Nichol (2021) proposed to change the second line of this equation and add a scaled gradient with respect to the target class from an externally trained classifier $c(\cdot)$ directly to the output of the denoising model:

$$\mathbf{x}_{t-1} \leftarrow \text{sample from } \mathcal{N}\left(\mu + s\Sigma\nabla_{\mathbf{x}_t} c(\mathbf{x}_t), \Sigma\right), \tag{6.12}$$

where s is a gradient scale.

With the joint training of a classifier and diffusion model introduced in this work, we propose to simplify the classifier guidance technique. Using the alternative training introduced in the previous section, Section 6.5.2, we can use noisy classifiers to formulate conditional sampling. The encoder model e_ν encodes input data $\mathbf{x}_t$ into the representation vectors $\mathcal{Z}_t$ that are used to both denoise an example into the previous diffusion timestep $\mathbf{x}_{t-1} \sim d_\psi(\mathcal{Z}_t)$ as well as to predict the target label with a classifier $\hat{y} = g_\omega(\mathcal{Z}_t)$. Therefore, to guide the model towards a target label during sampling, we propose optimising the representations $\mathcal{Z}_t$ according to the gradient calculated through the classifier with respect to the desired class. The overview of this procedure is presented in the Algorithm 1.

Algorithm 1 Sampling with optimised representations given a diffusion model (an encoder $e_\nu(\mathcal{Z}_t|\mathbf{x}_t)$, a decoder $d_\phi(\mathbf{x}_{t-1}|\mathcal{Z}_t)$), a classifier $g_\omega(y|\mathcal{Z}_t)$, and a step size α.

Input: class label y, step size α
$\mathbf{x}_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mathcal{Z}_t \leftarrow e_\nu(\mathbf{x}_t)$
  $\mathcal{Z}'_t \leftarrow \mathcal{Z}_t - \alpha\nabla_{\mathcal{Z}_t} \log g_\omega(y|\mathcal{Z}_t)$
  $\mu, \Sigma \leftarrow d_\psi(\mathcal{Z}'_t)$
  $\mathbf{x}_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu, \Sigma)$
end for
return $\mathbf{x}_0$

For the reformulation of the diffusion model proposed by Ho et al. (2020) where instead of predicting the previous timestep $\mathbf{x}_{t-1}$ denoising model is optimised to predict noise ϵ that is subtracted from the image at the current timestep $\mathbf{x}_t$, we adequately change the optimisation objective. Instead of optimising the noise to be spe-

cific to the target class, we optimise it to be *anything except for the target class*, which we implement by changing the optimisation direction: $\mathbb{z}_t' \leftarrow \mathbb{z}_t + \alpha \nabla_{\mathbb{z}_t} \log g_\omega(y|\mathbb{z}_t)$.

6.6. Experiments

In the experiments, we aim for observing the benefits of the proposed joint diffusion model over a stand-alone classifier or a marginal diffusion model. To that end, we run a series of experiments to verify various properties, namely:

- We measure the quality of a classifier to evaluate whether training together with a diffusion model improves the robustness of the classifier.
- We measure the generative capability of our model to check if representations optimised by the classifier can lead to more accurate conditional generations.
- We train our model in a semi-supervised setup to see if shared representations between the classifier and the diffusion model can positively influence the classification accuracy for a limited number of labelled data.
- We use a domain-adaptation task to check if optimising the representations using our approach helps to adapt to new data compared to a stand-alone classifier.
- We show that our joint model learns abstract features that can be used for the counterfactual explanation.

We use a UNet-based model with a depth level of three in all experiments. We pool its latent features with average pooling into a single vector, on top of which we add a classifier with two linear layers and the LeakyReLU activation. All metrics are reported for the standard training with the objective in (6.8), except for the conditional sampling where we additionally train the classifier on noisy samples, i.e., additional losses as in (6.10). Hyperparameters and training details are included in the appendix and code repository[1].

6.6.1. Predictive performance of joint diffusion models

In the first experiment, we evaluate the predictive performance of our method. To that end, we report the accuracy of our model on four datasets: FashionMNIST, SVHN, CIFAR-10, and CIFAR-100. We compare our method with a baseline classifier trained with a standard cross-entropy loss and the MLP classifier trained on top of representations extracted from the pre-trained DDGM as in Section 6.4, and three joint (hybrid) models: VERA (Grathwohl et al., 2021), JEM++ (Yang and Ji,

[1] `https://github.com/KamilDeja/joint_diffusion`

2021), HybViT (Yang et al., 2022b). The results of this experiment are presented in Table 6.6.1.

As noticed before, a classifier trained on features extracted from the UNet of a DDGM pre-trained in an unsupervised manner achieves reasonable performance. However, it is always outperformed by a stand-alone classifier. Interestingly, the proposed joint diffusion model achieves the best performance on all four datasets. The reason for that could be two-fold. First, training a partially shared neural network (i.e., the encoder in the UNet architecture) benefits from the unsupervised training, similarly to how the pre-training using Boltzmann machines benefited finetuning of deep neural networks Hinton et al. (2006). Second, the shared encoder part is more robust since it is used in the backward diffusion for images with various levels of noise.

Table 6.6.1. The classification accuracy calculated on the test sets. For each training of our methods and the vanilla classifier, we used exactly the same architectures. We report original reported results from related methods.

Model	F-MNIST	SVHN	CIFAR-10	CIFAR-100
VERA (Grathwohl et al., 2021)	-	96.8%	93.2%	72.2%
JEM++ (Yang and Ji, 2021)	-	96.9%	94.1%	74.5%
HybViT (Yang et al., 2022b)	-	-	95.9%	77.4%
Classifier	94.7%	96.9%	94.0%	72.3%
Ours (pre-trained DDGM)	60.6%	79.6%	80.9%	45.9%
Ours	**95.3%**	**97.4%**	**96.4%**	**77.6%**

6.6.2. Generative performance of joint diffusion models

In the second experiment, we check how adding a classifier in our joint diffusion models influences the generative performance. We use the FID score to quantify the quality of data synthesis. Additionally, we use distributed Precision (Prec), and Recall (Rec) for assessing the exactness and diversity of generated samples Sajjadi et al. (2018). For our joint diffusion model, we consider samples from the prior let through the backward diffusion. We also use the second sampling scheme in which we use conditional sampling, namely, the optimisation procedure as described in Section 6.5.3. We compare our approach with a vanilla DDGM, and a DDGM with classifier guidance Dhariwal and Nichol (2021), and recent state-of-the-art joint (hybrid) models: VERA Grathwohl et al. (2021), JEM++ Yang and Ji (2021), HybViT and GenViT Yang et al. (2022b).

Table 6.6.2. An evaluation of generative capabilities by measuring the FID score, Precision and Recall of generations from various diffusion-based models, including our joint diffusion model.

Model	FashionMNIST			CIFAR-10			CIFAR-100			CelebA		
	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑
DDGM	7.8	**71.5**	**65.3**	7.2	64.8	61.2	29.7	**70.0**	47.8	5.6	66.5	**58.7**
DDGM (classifier guidance)	7.9	66.6	59.5	8.1	63.2	**63.3**	22.1	69.3	46.9	4.9	66.0	57.8
Ours	8.7	71.1	61.1	7.9	69.9	56.4	17.4	63.2	54	7.0	**67.5**	51.5
Ours (conditional sampling)	**5.9**	63.1	63.2	**6.4**	**70.7**	54.3	**16.8**	63.5	**54.1**	**4.8**	66.3	56.5

Overall, our proposition outperforms standard DDGMs regarding the general FID, see Table 6.6.2. However, in some cases, the vanilla DDGM and the DDGM with the classifier guidance obtain better results in terms of the particular components: Precision (FashionMNIST, CIFAR-100) or Recall (FashionMNIST, CelebA). We can observe that conditional sampling improves the quality of generations in all evaluated benchmarks, especially in terms of precision that can be understood as the exactness of generations. This could result from the fact that the optimization procedure drives $\mathcal{Z}_t$ to a mode. Eventually, the backward diffusion generates better samples. However, comparing our approach to current state-of-the-art joint models, we clearly outperform them all, see Table 6.6.3.

Table 6.6.3. A comparison of generative capabilities of joint models by measuring the FID score.We report original reported results from related methods.

Model	CIFAR-10	CIFAR-100	CelebA
	FID ↓	FID ↓	FID ↓
VERA (Grathwohl et al., 2021)	27.5	-	-
JEM++ (Yang and Ji, 2021)	37.1	-	-
HybViT (Yang et al., 2022b)	26.4	33.6	-
GenViT (Yang et al., 2022b)	20.2	26.0	22.07
Ours	7.9	17.4	7.0
Ours (conditional sampling)	**6.4**	**16.8**	**4.8**

To get further insight into the role of conditional sampling, we carried out an additional study for the varying value of α (the step size in Algorithm 1). In Figure 6.6.1, we present how Precision and Recall change for different values of this parameter. Apparently, increasing the step size value α leads to more precise but less diverse samples. This is rather intuitive behaviour because larger steps result in features $\mathcal{Z}_t$ closer to modes. There seems to be a sweet spot around $\alpha \in [100, 250]$ for which both measures are high.

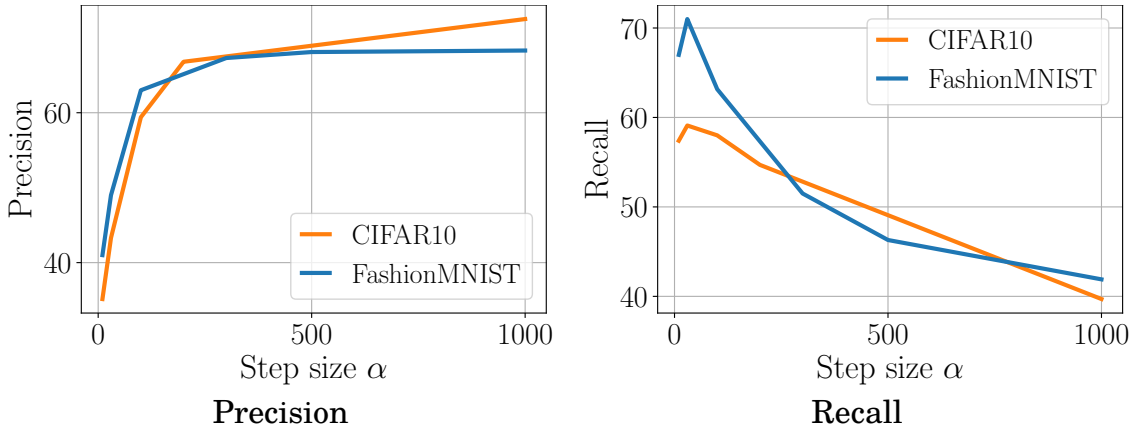


Figure 6.6.1. The dependency between the value of the step size α and the value of Precision and Recall for the joint diffusion with conditional sampling.

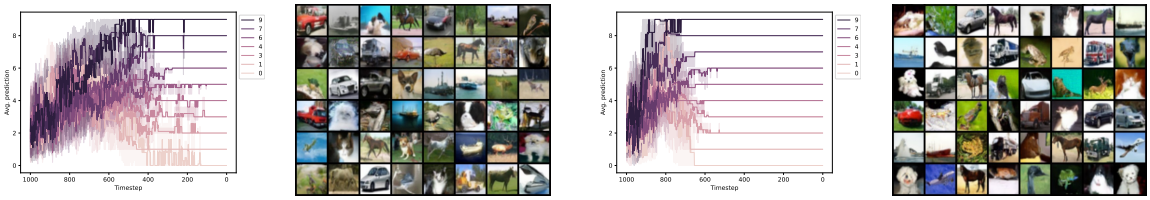
We visualise this effect in Figure 6.6.2. For a chosen class, e.g., plane, we observe that the larger α, the more precise the samples are but with limited diversity (i.e., the background is almost the same). For more samples, see Appendix 6.8.3.

In Fig. 6.6.3 we present how the decision of the classifier changes for sampling with the optimised generations. With a higher α step size value, optimisation converges faster towards target classes. Interestingly, for the CIFAR10 dataset, there are certain classes (e.g., class 3) that converge later in the backward diffusion process than the others. We also present associated samples from our model. Once more, they depict that higher values of the α parameter lead to more precise but less diverse samples. We show more generations from our joint model in the Appendix 6.8.3.

Figure 6.6.2. Samples from our joint diffusion model optimised towards a specific class (here: *plane*) with different step size α.

6.6.3. A comparison to state-of-the-art approaches

To get a better overview of the performance of our joint diffusion model, we present a comparison with other joint models and SOTA discriminative and gener-



(a) $\alpha = 200$ *(b)* $\alpha = 1000$

Figure 6.6.3. CIFAR10: Classifier decisions at different diffusion steps, for conditional sampling with different values of step size α and associated conditional samples

ative models in Table 6.6.4. Importantly, we present the discriminative model and the generative model as the bounds of the performance. The purely discriminative and generative models are included as the upper bounds of the performance. Importantly, within the class of the joint models, our joint diffusion clearly outperforms all of the related works.

Table 6.6.4. A comparison of our joint diffusion model with other joint models, and the SOTA discriminative model, and the SOTA generative model on the CIFAR-10 test set.

Class	Model	Accuracy% ↑	FID↓
Joint	IGEBM (Du and Mordatch, 2019)	49.1	37.9
	Glow (Kingma and Dhariwal, 2018)	67.6	48.9
	Residual Flows (Chen et al., 2019)	70.3	46.4
	JEAT (Grathwohl et al., 2019a)	85.2	38.2
	JEM (Grathwohl et al., 2019a)	92.9	38.4
	VERA ($\alpha = 100$) (Grathwohl et al., 2021)	93.2	30.5
	JEM++ (Yang and Ji, 2021)	94.1	38.0
	HybViT (Yang et al., 2022b)	95.9	26.4
	Ours	**96.4**	**7.9**
Disc.	VIT-H (Dosovitskiy et al., 2020)	99.5	-
Gen.	DDGM (our implementation)	-	7.2
	LSGM (Vahdat et al., 2021)	-	2.1

6.6.4. Semi-supervised learning of joint diffusion models

With satisfactory performance, we further evaluate other setups where one part of the model can benefit from another. In particular, we propose to assess our approach in the semi-supervised setup, where we artificially limit the amount of labelled data to 10%, 5% or 1% in three datasets SVHN, CIFAR-10, and CIFAR-100. We compare joint diffusion models to a deep neural network-based classifier and a

deep neural network-based classifier on top of the pre-trained UNet encoder. The results are presented in Table 6.6.5.

In the case of the stand-alone classifier, we observe that classification accuracy drastically drops with the number of labelled data. However, in our joint diffusion model, we can train the classifier on the smaller dataset while still optimising the generator part in an unsupervised manner, with all available unlabelled data. This approach significantly improves the classifier's performance thanks to the improved quality of data representations. For CIFAR-10, we observe that the joint diffusion model with only 5% of labelled data (250 examples per class) performs almost as well as the stand-alone classifier trained with the fully labelled training dataset. In more extreme scenarios, e.g., labelled data limited to 50, 25, or 5 examples per class, it seems to be slightly more beneficial to first learn the data representation in an unsupervised way and then add the classifier on top of them. However, overall, the joint diffusion model performs extremely well and greatly benefits from available unlabelled data in terms of classification accuracy. Our experiments align with the observation by Baranchuk et al. (2021), where DDGMs were used to improve the performance in semi-supervised image segmentation.

Table 6.6.5. The accuracy of the classifier trained in the semi-supervised setup, for each dataset we train the classifier with the fully labelled data or a limited amount of labelled examples and the remaining unlabelled examples. We compare standard classifier with classifier trained on a pre-trained DDGM as presented in Sec 6.4 and our joint diffusion method.

	SVHN			CIFAR-10			CIFAR-100			
Labelled data	100%	5%	1%	100%	5%	1%	100%	10%	5%	1%
Images per class	10000	500	100	5000	250	50	500	50	25	5
Classifier	95.1	87.8	75.15	81	46.4	31.5	60.8	22.2	16.6	6.9
Ours (pre-trained DDGM)	79.6	51.7	66.0	80.9	75.1	**65.3**	43.8	33.9	**28.8**	**15.4**
Ours	**95.4**	**90.2**	**76.7**	**89.9**	**78.2**	64.7	**63.6**	**38.6**	21.5	11.5

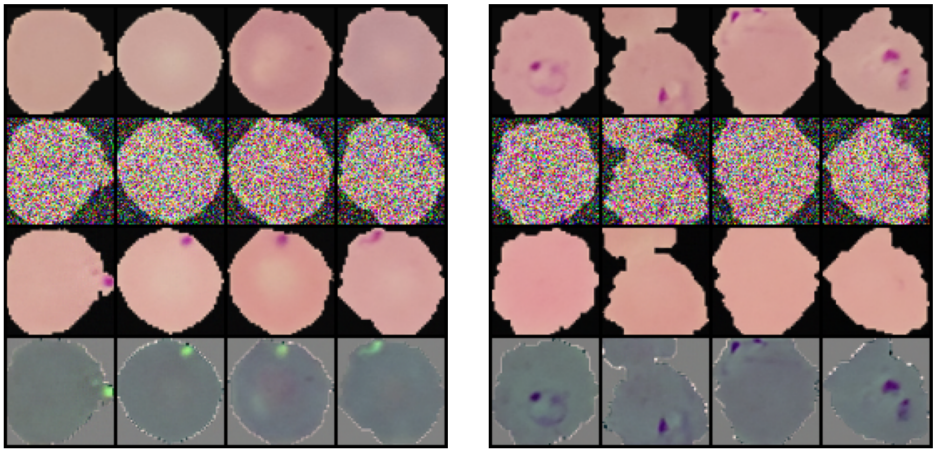
6.6.5. Domain adaptation with diffusion-based fine-tuning

In the previous section, we evaluate whether the classifier can benefit from the generative part of our model when trained with limited access to labelled data. Now, we further extend those experiments and check if joint diffusion can adapt to the new data domain using only the generative part – in a fully unsupervised way. For this purpose, we run an experiment in which we first train the model on the source labelled data to retrain it on the target dataset without access to the labels. We compare our approach to a standalone deep neural network-based classifier, see Table 6.6.6.

Table 6.6.6. The classification accuracy of the classifier trained in a domain adaption task. We first train the joint model on the source dataset, which we adapt to the target domain by retraining it using only the diffusion loss for the examples in the target one.

	SVHN → MNIST	USPS → MNIST	MNIST → USPS
Classifier	78.8	54.7	72.2
Ours	85.5	90.5	92.7
Classifier on target (upper bound)	96.1	96.8	99.4

As expected, in all three scenarios, the classification accuracy of the stand-alone classifier degrades on a target domain.[2] However, having access to unlabelled data from the target domain allows our joint diffusion model to adapt surprisingly well. Our approach outperforms the stand-alone classifier in all three cases by a significant margin. This result indicates that learning low-level features is essential for obtaining good predictive power while it is enough to transfer the classification head unchanged.



Negative examples Positive examples

Figure 6.6.4. Data samples from the Malaria dataset classified as negative examples (left) or parasitised cells (right). (*top row*) original data examples, (2^{nd} *row*) data noised with 20% of forward diffusion steps, (3^{rd} *row*) denoised images with conditional sampling, (*bottom row*) the difference between the 3^{rd} and 4^{th} rows.

6.6.6. Visual Counterfactual Explanations

In the last experiment, we apply our joint diffusion model to real-world medical data, the MALARIA dataset Rajaraman et al. (2018), that includes 27,558 cell images that are either infected by the malaria parasite or not (a classification task).

[2] The classification accuracy does not drop to a random level because all datasets share the same task, i.e., digits classification.

The cells have various shapes and different staining (i.e., colours) and contain or not the parasite (visually apparent as a purple dot).

After training our joint diffusion model, we obtain high classification accuracy (98%) on the test set. On top of this, we introduce an adaptation of visual counterfactual explanations (VCE) method Augustin et al. (2022) that provides an answer to the question: *What is the minimal change to the input image* $\mathbf{x}_0$ *to change the decision of the classifier*. In our setup, we answer this question with a conditional sampling algorithm that we use to generate the counterfactual explanations. In Figure 6.6.4, we show a few examples from the negative (left) or positive (right) classes. We add 20% of noise to these images and run conditional sampling with the opposite class (i.e., changing negative examples to positive ones and *vice versa*). In both cases, the joint diffusion model with conditional sampling can either remove the parasite from the image (for the positive examples) or add the parasite to the image (for the negative ones). All presented images are not cherry-picked.

This experiment shows that not only we can use our proposed approach to obtain a powerful classifier but also to visualise some regions of interest. In the considered case, calculating the difference between the original example and the image with a changed class label indicates the malaria plasmodium (see the last row in Figure 6.6.4). We provide more examples from the CelebA data in the Appendix 6.8.4.

6.7. Conclusion

In this work, we introduced a joint model that combines a diffusion model and a classifier through shared parameterisation. We first experimentally demonstrated that DDGMs learn semantically meaningful data representations that could be used for classification. On top of this observation, we introduced our joint diffusion models. In the experimental section, we showed that our approach improves the performance in both the classification and generative tasks, providing high-quality generations and enabling conditional generations with built-in classifier guidance. Our proposed approach achieves state-of-the-art performance in the class of joint models. Additionally, we show that the joint diffusion model can be used in semi-supervised learning, domain adaptation, and for counterfactual explanations, without any changes to the original setup.

6.8. Appendix

6.8.1. Training details and hyperparameters

Pooling of the UNet features As discussed in Section 6.4, we pool the UNet features encoded to different UNet levels with the average pooling function. Precisely speaking, we take an average convolutional filter activation for a given filter across the whole image. This approach seems to result in the loss of information, such as the location of particular features extracted by the convolutional filter, but it allows us to create image representation with reasonable dimensionality. Depending on the dimensionality of input, with our method, we extract 1856 features for 28×28 Gray-scale images (e.g. MNIST), 3712 features for 32×32 images with 3 colour channels (e.g. CIFAR), and 5248 features for 64×64 images with 3 colour channels (CelebA).

In all of our experiments, we use average pooling. Although other options such as max or min pooling might be used, our approach ensures that all of the features across the whole image are shared between the classifier and the generative models.

Semi-supervised learning In our semi-supervised learning, we train our joint diffusion model on datasets with limited access to labelled samples. The simplest approach for this problem is to calculate the loss function on the diffusion using the whole batch of data while using only the labelled examples for the classifier loss. However, in some scenarios, we artificially omit up to 99% of labelled data. In practice, this would lead to a situation where for batch size equal to 128 or 254 examples, the classifier loss would be practically calculated on 1 or 2 samples. Therefore, to stabilise the training we propose to create a buffer where we put labelled examples from each batch. When the buffer reaches its capacity equal to the batch size, we calculate the classifier loss using the examples from the buffer and add it to the generative loss according to Equation 6.8.

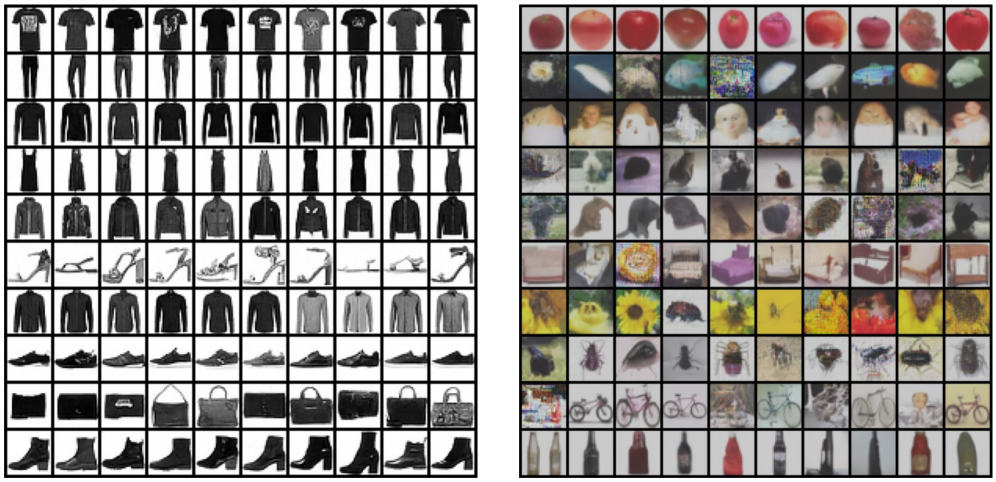
6.8.2. Domain adaptation

In the experiments on the domain adaptation task, we propose the simplest setup, where we first train the joint model on the source task using the joint loss function (Eq. 6.8), and then we retrain the model on the target domain using only the DDGM loss in Equation 6.4. We show that without any alteration to our basic setup, we can observe a significant performance boost compared to the baseline classifier. We believe that we can further improve those results if we focus directly on the domain adaptation task and take advantage of the recent advantages in this

field. Further experiments in this direction should for example include simultaneous training on examples from two domains. To improve the alignment, we can also benefit from adversarial training as introduced by Ganin et al. (2016) in DANN.

6.8.3. Additional results: Conditional generations with optimised representations

In Fig. 6.8.1 and Fig. 6.8.2 we present additional generations from our joint--diffusion model sampled with and without conditional sampling technique. Our method is capable of generating high quality samples that are precise and diverse.



Fashion MNIST CIFAR-100

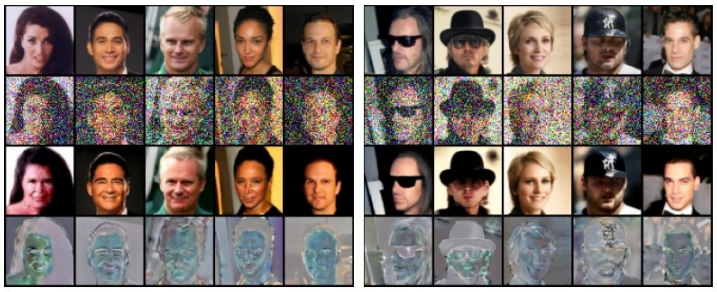
Figure 6.8.1. Conditional samples from our joint diffusion model for Fashion MNIST dataset (*left*) and first 10 classes of CIFAR100 dataset (*right*). Each row represents samples from one class.

Fashion MNIST CIFAR-100 CelebA

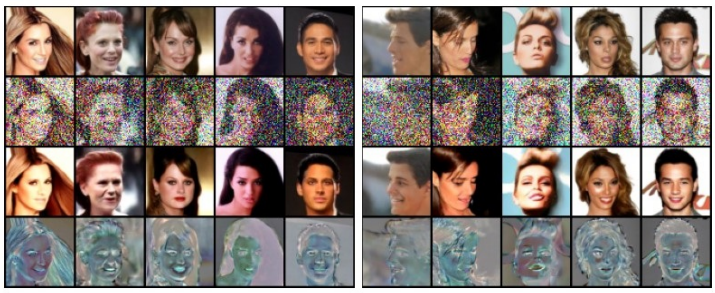
Figure 6.8.2. Generated examples from our joint diffusion model without conditional sampling for CIFAR-10, CIFAR-100, and CelebA dataset.

6.8.4. Additional results: Counterfactual image generation

In the experiment described in Section 6.6.6, we presented how we can use our joint diffusion model to generate the counterfactual explanations to the classifier using the medical dataset. In Figure 6.8.3, we present more examples of this approach by perturbing original examples from the CelebA dataset. We select 3 attributes from the CelebA dataset namely: *young*, *smiling*, and *moustache*. For each attribute, we select 5 positive examples and 5 negative examples which we alter using our conditional sampling procedure with the classifier-based optimisation. We present original examples (first row) noised with 20% of noise (second row) and generated towards counterfactual class (third row). In the last row, we show the differences between the original and modified examples.



(a) Young to old (*left*), old to young (*right*)



(b) Smiling to no-smiling (*left*), no-smiling to smiling (*right*)

(c) Moustache to no-moustache (*left*), no-moustache to moustache (*right*)

Figure 6.8.3. Counterfactual image generation for the CelebA dataset using three different attributes on random original examples. For each attribute, we select 5 positive examples that we change to negative ones and 5 negative ones that we change to positive ones.

7. Background – Continual Learning

So far, we have considered generative models in the most common stationary setup, where a model is trained only once, using available training data, and validated with a stationary test set of similar data distribution. However, in many real-world scenarios, the training data for machine learning systems come in separate portions (known as *tasks*). For example, an autonomous driving system trained over the data acquired in certain weather conditions (e.g. in summer) should be retrained when a new portion of data arrives (e.g. with snowy winter conditions). In such a scenario, we would expect the model to learn from additional winter data samples while maintaining its performance on sunny days. Unfortunately, recent neural methods based on training with gradient-based optimisation suffer from the problem known as *catastrophic forgetting* (French, 1999) – an abrupt loss in performance on the previous portion of data while retrained on the new one. Therefore, the most common practical solution for training a model on a data stream is to retrain it from scratch every time a new portion of data arrives, using all the data gathered so far. Such an approach brings several apparent drawbacks, as it requires storing all the historical data samples and computationally expensive retraining whenever a new portion of data is available. *Continual Learning* (CL) methods are designed to overcome those shortcomings arising when training machine learning models with a continuous stream of data.

In particular Hadsell et al. (2020) and Pascanu (2021) distinguish the five most important desiderata for continual learning systems:

- **Minimal access to previous tasks.** Ideally, the CL system should be able to work infinitely. Therefore it should not require growing storage capacity for previous data samples.
- **Minimal increase in model capacity and computation.** The solution should be scalable. We cannot simply add a new model with every task
- **Minimising catastrophic forgetting**. Training on a new task should not result in a significant loss in performance on previously learned tasks.
- **Maintaining plasticity.** Preventing forgetting cannot be achieved at the expense of the performance on the new data. The model should be able to effectively learn from new tasks.

- **Maximising forward and backward transfer.** Learning from one task should facilitate learning from similar past and future tasks.

7.1. Continual Learning Methods

The problem of continual learning, also known as lifelong learning, attracts the increasing attention of the research community (Parisi et al., 2019; Mundt et al., 2020a; Qu et al., 2021; Mundt et al., 2023), which results in a growing number of novel methods that try to fulfil the above-mentioned objectives with different approaches. Usually, CL methods are divided into three categories which we overview in this section.

7.1.1. Methods based on regularisation

The regularisation-based CL approaches aim to strike a delicate balance between preserving previously acquired knowledge and providing sufficient flexibility to incorporate new information. The need for such a balance arises due to the tension between two competing goals – maintaining stable performance on previously learned tasks and rapidly adapting to the new ones (i.e. accommodating new knowledge) known as *stability-plasticity dilemma* (Hebb, 1949). Models trained without continual learning methods have very high plasticity and no stability. They can quickly adapt to new data distribution by completely ignoring previously encoded knowledge. Although we sometimes befit from this property (e.g. in transfer learning), it is usually a significant drawback of recently used methods that forget previously encoded knowledge catastrophically. Therefore, to ensure stability, regularisation-based CL methods apply some additional penalty that slows the process of overwriting previously learned knowledge.

In Elastic Weights Consolidation (EWC) (Kirkpatrick et al., 2017a) propose finding weights that are crucial for previous tasks using Fisher information to slow down their updates. Similarly, in Synaptic Intelligence (SI) (Zenke et al., 2017a) and Memory Aware Synapses (MAS) (Aljundi et al., 2018), additional information is stored together with each neuron in order to assess their significance. Several works employ Bayesian theory (Bayes, 1763) to estimate uncertainty and use it to either regularise weights (Ahn et al., 2019) or to slow down their updates by adequately adapting the learning rate (Ebrahimi et al., 2019). Simultaneously, a family of regularisation approaches is inspired by the knowledge distillation technique originally proposed by Hinton et al. (2015) for the compression of neural networks. For example, in Learning Without Forgetting (LWF) (Li and Hoiem, 2017), the con-

tinually trained model is regularised by minimising the difference in predictions for the samples from a novel task processed by a new model and its copy frozen at the beginning of a new task. The extension of this method combined with separated softmax is introduced by Ahn et al. (2021).

In general, regularisation-based CL approaches offer a promising avenue for mitigating the problem of catastrophic forgetting in neural networks trained for multiple tasks. They directly tackle the problem without relying on additional resources. Nevertheless, the performance of this family of methods in complex and diverse scenarios is limited.

7.1.2. Methods based on dynamic architectures

Hence, methods based on dynamic architectures build different model versions for different tasks to improve its performance in a multitask setup. Usually, a single neural model is split into a shared part used in every task and task-related submodules. During inference, the sample is first assigned to the proper task (or their combination) and then inferred through the specific part of a network. In Reinforced Continual Learning (RCL) Xu and Zhu (2018), Dynamically Expandable Networks (DEN) Yoon et al. (2018), and Progressive Neural Networks (PNN) Rusu et al. (2016) new structural elements are added to the model for each new portion of data. Alternatively, in Masse et al. (2018); Mallya and Lazebnik (2018); Golkar et al. (2019); Mallya et al. (2018), authors introduce methods for selecting different submodels of the larger model that are used only for consecutive tasks. Some architecture-based methods focus on specific components of the model. E.g. Yan et al. (2021) propose to train a new task-specific feature extractor, while on the other hand, Zhang et al. (2021) introduce continually evolved classifiers.

In general, methods in this category provide high accuracy in a task-incremental scenario when test samples are presented together with a corresponding task index Van de Ven and Tolias (2019). Otherwise, the task index needs to be estimated. Current approaches solve this issue using heuristics, such as the minimisation of classification entropy Hendrycks and Gimpel (2017); Wortsman et al. (2020), or with additional structures like gating autoencoders (Aljundi et al., 2017).

Apart from an explicit split in the model's architecture, a branch of works implicitly divides the capabilities of neural networks through orthogonalisation of the directions in which different tasks are processed. In particular Zeng et al. (2019), propose an Orthogonal Weights Modification (OWM) algorithm, where the modifications of weights in different tasks are mapped onto a selected subspace that does not affect previous tasks. Similarly, Wang et al. (2021c) benefit from the fact that models trained in a supervised manner rarely utilise the full network capabilities,

but in fact, every layer usually applies low-dimensional transformations. Therefore, the authors introduce the Adam-NSCL algorithm, which maps the original gradient values into the null spaces of the previous tasks.

7.1.3. Methods based on replaying

While methods gathered around the previous two groups focus on preventing the forgetting of continually trained models, methods based on replaying are based on the assumption that forgetting is inevitable and can only be overcome by a constant rehearsal with previous data examples. Therefore, the techniques described in this section preserve some form of previous data and use it alongside new samples when retraining the model.

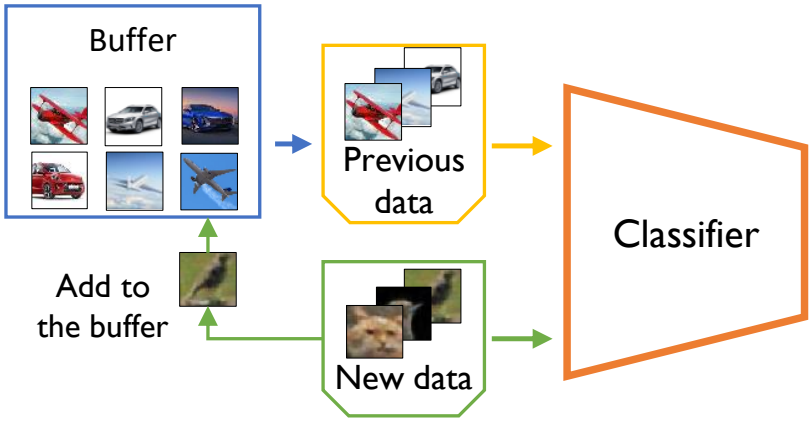


Figure 7.1.1. The schematic visualisation of rehearsal based continual learning system, where images from past tasks are stored in the examples buffer and added to the new data examples when retraining the model.

The first group of works falling into this category, including Rolnick et al. (2019); Aljundi et al. (2019), employ a memory buffer to store randomly selected or possibly most relevant previous data examples. We present the schematic overview of this approach in Figure 7.1.1. On top of this basic schema, different techniques are used to select relevant data samples that should be stored for further rehearsal. In particular, Isele and Cosgun (2018) introduce Selective Experience Replay (SER), where exemplars are stored according to the carefully balanced trade-off between those favouring surprise or reward. Alternatively, GEM/A-GEM (Lopez-Paz and Ranzato, 2017a; Chaudhry et al., 2019) combine the idea of rehearsal from examples with constraints on the update gradients. To further simplify the rehearsal Belouadah and Popescu (2019) point out that storing task-related statistics can be beneficial for rehearsal. Therefore they propose the Incremental Learning with Dual Memory

approach, which holds two different types of data related to previous tasks. Finally, Prabhu et al. (2020) introduce a simple approach called *GDumb*, highlighting the limitations of recent CL methods. Authors surprisingly show that retraining a model from scratch from a small memory buffer can form a solid benchmark that outperforms several CL techniques.

Although storing data samples in a memory buffer allows accurate results, at least several examples from each incoming task must be remembered. This requirement makes the solution inadequate for the general continual learning problem, in which we would like to retrain the model in a potentially infinite number of tasks.

The most straightforward approach to overcome this burden was proposed already by Robins (1995), where so-called *pseudo-rehearsal* random samples were used to prevent forgetting. This idea has changed significantly with the introduction of Deep Generative Replay (Shin et al., 2017), where the buffer was replaced with a generative model based on the Generative Adversarial Networks (GAN). However, since any structure used to compress past data may also suffer from catastrophic forgetting, the authors propose a self-rehearsal procedure to train the generative model with data from new tasks and regenerated examples from the previous ones. This approach is used in the majority of generative rehearsal-based CL methods. We present the schematic overview of generative rehearsal in Figure 7.1.2.

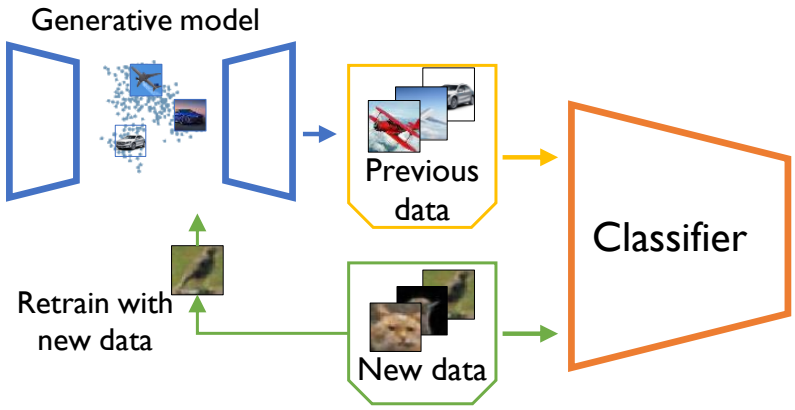


Figure 7.1.2. The schematic visualisation of generative rehearsal based continual learning system. An additional generative model is used that learns to generate past data samples as rehearsal example that are added to the new data batch when retraining the model.

Similarly to Shin et al. (2017), van de Ven and Tolias (2018) introduces a generative rehearsal model based on Variational Autoencoder (VAE). Additionally, they combine the generative model with the base classifier to reduce the cost of model retraining. Similarly, Scardapane et al. (2020) combines classifier with a normalising flow model, which they additionally regularise with respect to the past tasks, while Rostami et al. (2019) employ Gaussian Mixture Model (GMM) as a source of

rehearsal examples. VAE was also used in favour of GANs by Mundt et al. (2020b), who successfully incorporated this model into the problem of continual learning with open datasets. More generally, Lesort et al. (2019) overview how different generative models behave in continual learning setup. Their analysis includes different generative autoencoders and several versions of GANs. The general conclusion of this work is that the highest performance in terms of image synthesis can be observed with GANs. However, all the methods trained with the Generative Replay strategy struggle with more complex datasets.

Therefore, apart from methods that aim to generate original input from the previous tasks, a branch of works focuses on rehearsing the internal data representations instead, as presented in Figure 7.1.3. This idea was introduced in Brain Inspired Replay (BIR), where Van de Ven et al. (2020) propose to replay data latent representations obtained from a first few layers of a classifier, commonly known as *feature extractor*. For simple datasets, authors propose to train the feature extractor alongside the classifier, while for more complex ones, they pre-train and freeze the first few convolutional layers. This depicts the main challenge in this group of methods – they have to assume that the difference between tasks is not substantial enough to spoil the replayed representations or the feature extractor has to be frozen after pre-training.

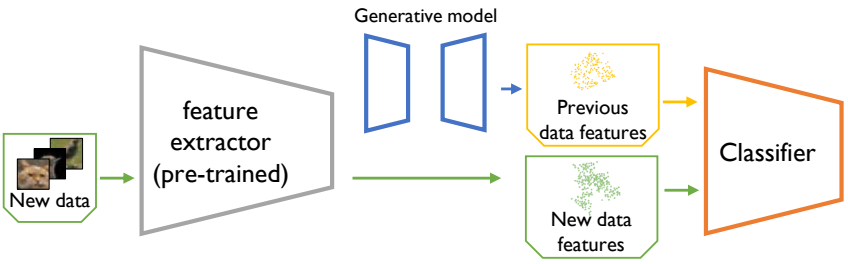


Figure 7.1.3. The schematic visualisation of feature-replay based continual learning system, where images from past tasks are stored in the memory buffer. When retraining the basic model (e.g. classifier) rehearsal samples selected from the buffer are added to the new data examples to prevent forgetting.

The idea of feature replay is further explored by Kemker and Kanan (2018), where feature replay is divided into short and long-term parts. Contrary to previous approaches based on VAE, Liu et al. (2020) explore feature replay with conditional generative adversarial networks. While those techniques provide state-of-the-art performance on complex datasets, they all utilise a frozen feature extractor, what limits the usability of the method. To overcome this drawback, Thandiackal et al. (2021) introduce Generative Feature-Driven Image Replay where authors combine image and features replay, which allows them to continually train (in a limited fashion) both the classifier and feature extractor.

7.2. Continual Learning of Generative Models

In all of the works described in the previous section, the generative model is employed as a source of previous data rehearsal examples to train a base classifier. However, it is known that any neural model can suffer from catastrophic forgetting. In particular, generative models continually retrained with a new task of radically different distribution, quickly adapt to the new one and completely forget how to generate samples from the previous ones. Therefore, a growing field of research focuses on continual learning of generative models beyond generative replay.

In particular Nguyen et al. (2018) adapt regularisation-based methods such as Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017a), and Synaptic Intelligence (SI) (Zenke et al., 2017a) to the continual learning in generative models regularising the adjustment of the most significant weights. The authors also introduce Variational Continual Learning (VCL), a dynamic architecture that adds task-specific modules to the variational autoencoder.

In VASE by Achille et al. (2018), authors propose a method for continual learning of shared disentangled data representation. While encoding images with a standard VAE, VASE also seeks shared generative factors. A similar concept of mixed-type latent space was introduced in LifelongVAE (Ramapuram et al., 2020), where it is composed of discrete and continuous values.

In HyperCL, von Oswald et al. (2019) propose an entirely different approach with a hypernetwork generating the weights of the continually trained model. This yields state-of-the-art results in discriminative models task-incremental training but also applies to the generative models. In particular, the authors introduce a method for generating variational autoencoder's weights. In order to differentiate tasks, Rao et al. (2019) introduce CURL that learns task-specific representation and deals with task ambiguity by performing task inference within the generative model. This approach directly addresses the problem of forgetting by maintaining a buffer for original instances of poorly-approximated samples and expanding the model with a new component whenever the buffer is filled. In BooVae, Egorov et al. (2021) propose an approach for continual learning of VAE with an additive aggregated posterior expansion. Several works train GANs in continual learning scenarios either with memory replay (Wu et al., 2018), with the extension to VAEGAN in Lifelong-VAEGAN by Ye and Bors (2020).

In our recent preliminary studies (Zajac et al., 2023), we investigate the problem of continual learning of deep generative diffusion models. We show that those powerful generative models suffer from catastrophic forgetting similarly to other models. We evaluate that classical CL methods can help prevent abrupt loss in

performance. However, we also observe interesting phenomena such as overfitting to the data stored in the memory buffer and the fact that the forgetting changes with diffusion timesteps.

7.2.1. Knowledge Consolidation with Generative Modelling

As a part of generative continual learning, a group of methods considers generative models as a focal point of a continually trained system. This assumption is based on neuroscience research that often serves as an inspiration for continual learning methods Hayes et al. (2021). In particular, the inspiration most commonly associated with generative replay relates to the role of a hippocampus – a part of the human brain that consolidates short-term memories into long-term ones during sleep (Müller and Pilzecker, 1900). Neurological studies (Dudai et al., 1985; Tempel et al., 1983) highlight that this process requires activity that extends beyond the duration of the learning process – stimuli that we can understand as an internal rehearsal. In fact, mechanisms similar to those used in neural networks (feedback and feedforward connections) are used across many memory-related processes, such as integration of internal brain states, re-evaluation, and updating of learned information and long-term memory consolidation (Krashes and Waddell, 2008; Colomb et al., 2009; Cognigni et al., 2018). Following this study, Stoianov et al. (2022) introduce a computational theory where the hippocampus is formulated as a hierarchical generative model with three layers that accumulate knowledge structured to a different level. As presented in (Stoianov et al., 2022), this theory explains the need for generative replay in the process of knowledge consolidation.

Following these considerations, Thai et al. (2021) evaluate the continual learning of reconstruction tasks showing that autoencoders are less prone to the problem of catastrophic forgetting. In our work (Masarczyk et al., 2021), we further explore this direction and evaluate the robustness of data representations learned with a generative model. Our experiments indicate that joint modelling significantly reduces the problem of catastrophic forgetting. Similarly, Rostami et al. (2019) propose to encode data into the latent space of the autoencoder and approximate its latent space with Gaussian Mixture Model. Such an approach allows accurate rehearsal while, at the same time, latent features can be used as input for the continually trained classifier.

In the same spirit, there exists a group of methods directly inspired by the human memory system. Kamra et al. (2017) introduces a Deep Generative Dual Memory Network, where two generative models are used to emulate the hippocampus and the neocortex. The first encodes short-term memory by rapidly learning a new task, which is slowly transferred to the second, a long-term memory model. This

idea is further enhanced into a triple-memory network (Wang et al., 2021b), where a system composed of a GAN with an additional independent classifier consolidates knowledge.

8. BinPlay: A Binary Latent Autoencoder for Generative Replay Continual Learning

Title	BinPlay: A Binary Latent Autoencoder for Generative Replay Continual Learning
Authors	Kamil Deja, Paweł Wawrzyński, Wojciech Masarczyk, Daniel Marczak, and Tomasz Trzciński
Conference	2021 International Joint Conference on Neural Networks (IJCNN)
Year	2021
DOI	10.1109/IJCNN52387.2021.9534171

Preface

In this work, we look at the latent data representation in generative models from yet another perspective – an efficient data compression mechanism. To that end, we introduce a method called BinPlay where a binary autoencoder is used as a method for efficient storage of data examples. To evaluate this approach, we use it in the continual learning setup, where generative models are employed to serve as a source of rehearsal examples for the continually trained classifier.

Contrary to recently proposed methods described in Chapter. 7.1.3, we observe that instead of using continuous data representations that allow for generation of new examples from a continuous data distribution, we can focus on several training examples that can be encoded into categorical values. In particular, we draw our attention to binary representations that are much more efficient in storage. In fact, we push this design to the limit and introduce a method for calculating binary codes that can be used for encoding data examples without a need for their memorisation.

Through a series of experiments we show that binary data representations in our BinPlay provide an efficient way for the storage of past examples. Moreover, we show that our autoencoder is able to learn from consecutive tasks and align different latent representations.

Abstract

We introduce a novel binary latent space autoencoder architecture to rehearse training samples for the continual learning of neural networks. The ability to extend the knowledge of a model with new data without forgetting previously learned samples is a fundamental requirement in continual learning. Existing solutions address it by regularising network weights, adjusting its architecture, or retraining with past data samples, regenerated from memory or reconstructed with generative models. Unfortunately, recreating past data from memory requires an infinite buffer, while the reconstructions of generative models tend to miss details of individual samples when generalising beyond the training set. In this paper, we aim to overcome these limitations and introduce a novel generative rehearsal approach called BinPlay. Its main objective is to find a quality-preserving encoding of past samples into precomputed binary codes living in the autoencoder's binary latent space. Since we parameterise the formula for precomputing the codes only on the training samples' chronological indices, the autoencoder is able to compute the binary codes of rehearsed samples on the fly without the need to keep them in memory. Evaluation on three benchmark datasets shows up to a twofold accuracy improvement of BinPlay versus competing generative replay methods.

8.1. Introduction

While mammal brains are capable of acquiring new knowledge without forgetting the skills learned in the past French (1999), artificial neural networks fail to copy this behaviour. The resulting *catastrophic forgetting* phenomenon French (1999) manifests itself in the deteriorating performance of a model on a previously learned task upon learning a new one. This is due to the fact that in the training phase, a neural network assumes a stationary data distribution that does not change in time. Hence, re-training the network with new input data adjusts its weights to improve the current objective's performance, disregarding previously seen samples and the associated knowledge Kirkpatrick et al. (2017b). The assumption of stationary data distribution often fails in practice, as numerous applications, including experimental physics Tilaro et al. (2018), autonomous driving Yang et al. (2018), robotics Kehoe et al. (2015), and recommendation engines He et al. (2017), observe ever-growing datasets with changing data characteristics.

A domain of machine learning that attempts to address these requirements is referred to as *continual learning*. Methods of continual learning proposed in the literature include regularisation of neural network weights Zenke et al. (2017b);

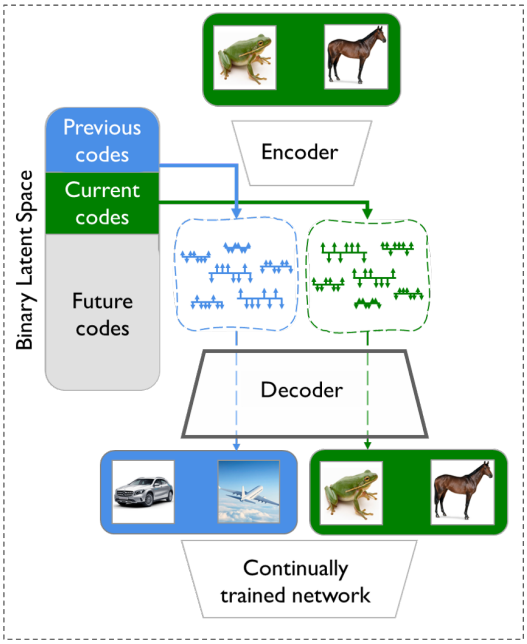


Figure 8.1.1. Overview of our BinPlay architecture for continual learning. When a new task arrives, we train the encoder to map new images to a set of precomputed binary codes in the latent space, and then we train the decoder to reconstruct them back to their originals. Since we compute the binary codes using only the indices of training data samples, we know which part of the latent space is populated with the codes of previously seen data. We can also re-compute those codes on the fly, using the indices, to rehearse past samples. This way, our BinPlay model can encode information about the past data in the autoencoder's binary latent space, without the need to store individual images or their codes in memory. Visualisation of the main concept of BinPlay is presented in the video: `https://youtu.be/0b_8q_rwxzg`.

Kirkpatrick et al. (2017b), adjustments in the structure of a network to the next task Xu and Zhu (2018); Yoon et al. (2018); Rusu et al. (2016); Golkar et al. (2019); Cheung et al. (2019); Wen et al. (2020), or replaying examples from previously seen tasks while training the network with new data Rebuffi et al. (2017); Lopez-Paz and Ranzato (2017b), also known as rehearsal training. As storing past data requires a growing buffer, recent methods for replaying past samples leverage generative neural architectures Goodfellow et al. (2014b); Kingma and Welling (2014) to reconstruct previously seen data points from various probability distributions van de Ven and Tolias (2018); Xiang et al. (2019); Mundt et al. (2020b).

These models, referred to as generative rehearsal methods, theoretically allow infinite continual learning without inflating past examples' buffer. However, their performance is upper-bounded by the quality of the training data recreated with generative models. Since the most frequently used generative models, such as Generative Adversarial Networks (GAN) Goodfellow et al. (2014b) and Variational Autoencoders (VAE) Kingma and Welling (2014), are trained to learn a continu-

ous function that describes the entire dataset distribution, they often yield visually plausible yet low-quality results. For more challenging datasets, generative models often output blurry images that miss high-frequency details Lesort et al. (2019); Aljundi et al. (2019). When used as samples for the rehearsal procedure, such examples lead to the deteriorated performance of neural network models Lesort et al. (2019).

In this paper, we address these shortcomings by introducing an autoencoder--based architecture named BinPlay[1] that is able to regenerate the samples seen in the previous tasks out of a set of precomputed binary codes. Inspired by generative hashing approaches Carreira-Perpinan and Raziperchikolaei (2015); Mena and Nanculef (2019); Zamorski et al. (2020), we achieve that goal by designing a binary latent space autoencoder that learns the most quality-preserving mapping between data samples from the current task and a set of binary vectors living in the autoencoder's latent space. We allocate the pool of binary codes using only a chronological ordering of training samples, *i.e.*, the ordinal number at which samples are presented to the model. Since we know the number of data samples seen in previous tasks, we can rehearse past images by first sampling binary codes uniformly from the pool of codes already *allocated* and then decoding them with the decoder. This way, we do not need to store any binary embeddings for previous tasks in memory. At the same time, we avoid the problem faced by competing generative rehearsal methods, namely the implicit requirement for the generative model to generalise beyond a set of training samples.

Fig. 8.1.1 illustrates the overview of our method. Its general goal is to make a neural network fit data that are given in subsequent tasks. Therefore, it can be used in any problem of continual data modelling. In this work, we evaluate our BinPlay on the image classification benchmarks against competitive approaches in the most challenging *class incremental* learning scenario. BinPlay consistently outperforms state-of-the-art methods on three benchmark datasets while providing a comparable or lower model memory footprint.

To summarise, the contributions of this work are:

- a new approach to continual learning problem called BinPlay that combines advantages of generative and buffer-based rehearsal methods by encoding previously seen samples not in memory but in the binary latent space of an autoencoder,
- a novel binary latent space autoencoder architecture which learns a quality--preserving, one-to-one mapping between original data samples and predefined binary codes,

[1] We make the code available at `https://github.com/danielm1405/BinPlay`

- a binary code assignment method based on the ordinal number of training images, which enables the computation of binary codes corresponding to rehearsal samples on the fly, without the need to store them in the memory.

The remainder of this paper is organised as follows. In the next section we overview, the related techniques, Sec. 8.3 introduces our proposed BinPlay approach and in Sec. 8.4 we present the experiments confirming validity of our method. We conclude this work in Sec. 8.5.

8.2. Related Works

This work falls within rehearsal Continual Learning methods described in Chapter 7.1.3. Our method regenerates previous examples from a neural model, but contrary to other generative rehearsal methods we do not aim in learning previous tasks data distribution. Instead, we can relate to the interesting idea that we can place in between generative and buffer based rehearsal presented in Caccia et al. (2020), where authors incorporate VQ-VAE van den Oord et al. (2017) architecture to compress the original data examples into a special representation that requires less memory than original images. We extend this idea, and introduce a method that compresses the data examples to the set of specific binary vectors, which, contrary to VQ-VAE, does not have to be stored in the buffer. Our approach is conceptually similar to that in Mundt et al. (2020b). We train an autoencoder with an increasing set of images to reproduce them for the base model's future training.

8.3. Method

In this paper, the problem we consider is to efficiently store a *base* neural network training data, which is coming in subsequent tasks. With those in place, on the arrival of a new task, we can retrain the base network on the combination of new data samples and recreations of previous tasks. To regenerate the old data, we propose a novel binary latent space autoencoder, an auxiliary neural network inspired by the hashing autoencoder approaches Carreira-Perpinan and Raziperchikolaei (2015); Mena and Nanculef (2019); Zamorski et al. (2020). Its objective is to find an assignment of the input data to a set of predefined binary codes and then reconstruct them back to the original data. Therefore, the autoencoder's decoder can regenerate past data just by transforming randomly sampled codes. In Fig. 8.3.1 we show an overview of this mechanism applied to the incremental image classification problem. The next subsections discuss our approach's building blocks

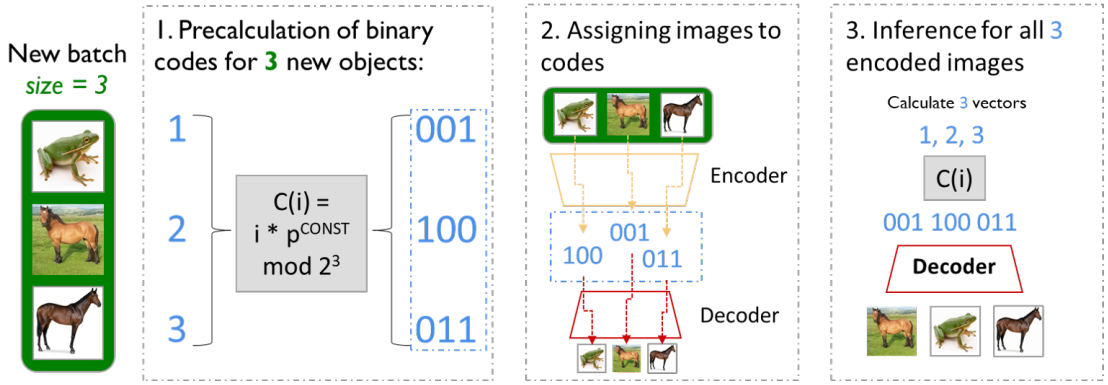


Figure 8.3.1. Encoding and decoding BinPlay binary latent space codes. We first create a set of binary codes on the basis of consecutive indices (**I**). We then train the encoder to find the best mapping between original inputs and a set of binary codes, and the decoder to reconstruct them back to their originals (**II**). To generate all previously encoded inputs, we only need the decoder and the total number of encoded samples (equal to the last sample's index, if counted from 1), so we can recompute the set of binary codes and process them through the decoder (**III**).

in more detail, namely the binary latent autoencoder, the definition of binary codes, and their assignment to input data.

8.3.1. Binary latent autoencoder

The objective of our autoencoder is to reconstruct a set of input points. Therefore, the encoder needs to learn their mapping to a set of predefined binary codes living in the autoencoder's latent space. The decoder, on the other hand, should be able to reproduce inputs from previous tasks based on the latent binary vectors assigned to them.

In order to train the binary latent autoencoder network to fulfil the above requirements, we follow the procedure introduced in Shin et al. (2017) and use both data from a new task and the sampled outputs from its current version. Thanks to this approach, we learn mappings and reconstructions for the new samples while preserving the knowledge of the previous ones at the same time. First, we create a copy, θ', of current decoder weights, θ, so that decoder with weights θ' can serve as a source of previous training pairs for the network.

For the training of our autoencoder, we create a set of training samples by uniformly drawing indices $i \in \{1,\ldots,K,\ldots,N\}$, where K is the first index for the current task, and N is the total number of data points observed so far.

- If i is between 1 and $K-1$, we generate the input-output pair $\langle c(i), p(c(i);\theta')\rangle$, where c is a binary codes generator function and p is our decoder.
- If i is between K and N, we use the new input-output pair $\langle c(i), \mathbf{x}_i\rangle$, where $\mathbf{x}_i$ is an original data input assigned to the index i.

With combination of such pairs, we adjust the original weights of our decoder θ which we train to reconstruct data inputs from their binary codes. To ensure that the above sampling procedure in the latent space works, we propose an appropriate binary code definition.

8.3.2. Binary codes definition

As we want to rehearse previously seen data, we require the set of binary codes within the latent space to be parameterised by an index. Then, in order to produce a set of uniformly sampled random binary codes that can be used to reconstruct the previously seen data, we only need to sample a set of indices. Our key requirement is that:

1. the codes are composed of a large number of bits
2. bits often change between the subsequent indices

This way, codes become easily recognizable inputs for the decoder network. To fulfill the above requirements, we follow an observation presented below.

Lemma 1. *Let λ be a prime number, $\lambda \neq 2$, and $i \in \{1,\ldots,2^m\}$ be an index coded as m least significant bits of the integer*

$$i \cdot \lambda^{\lfloor m \ln 2/\ln \lambda \rfloor}. \tag{8.1}$$

Then

1. $2^m/\lambda < \lambda^{\lfloor m \ln 2/\ln \lambda \rfloor} < 2^m$.
2. *The codes for all the numbers i are different.*

Proof: We have

$$\lambda^{\lfloor m \ln 2/\ln \lambda \rfloor} < \lambda^{m \ln 2/\ln \lambda} = \lambda^{(\log_\lambda 2)m} = 2^m,$$

and

$$\lambda^{\lfloor m \ln 2/\ln \lambda \rfloor} > \lambda^{m \ln 2/\ln \lambda - 1} = \lambda^{(\log_\lambda 2)m - 1} = 2^m/\lambda$$

which proves point 1.

In order to prove point 2, we notice that since λ is prime, and $\lambda \neq 2$, λ raised to any natural power is co-prime with 2^m. Suppose there are two different $i, j \in \{1,\ldots,2^m\}$ that have the same code in the form of m least significant bits of (1). That would mean that

$$(|i-j| \cdot \lambda^{\lfloor m \ln 2/\ln \lambda \rfloor}) \mod 2^m = 0.$$

That can not be true since $\lambda^{\lfloor m \ln 2/\ln \lambda \rfloor}$ has no common divisors with 2^m and $|i-j|$ is not divisible by 2^m because $|i-j| < 2^m$. ■

With the above observation, we define a coding of the sample index, $c : \mathbb{N} \mapsto \{0,1\}^n$, as follows:

1. The vector $c(i)$ is composed of several subvectors.
2. For a subvector of size m, we assign the m least significant bits of the binary representation of the formula (8.1) with i as the index, and different base prime numbers for different subvectors.

As a result of this procedure, the binary codes calculated are different and sparsely distributed. Moreover, as noticed in the Lemma 1, active bits of the codes calculated with the proposed procedure often change between subsequent codes. Hence they fulfil our requirements.

In a typical continual learning scenario, data samples within a task share some similarities, as they are constructed with the assumption of stationary data distribution. Therefore, we append a task index transformed using the procedure above as a prefix of the binary codes defined for a given task.

8.3.3. Binary codes assignment

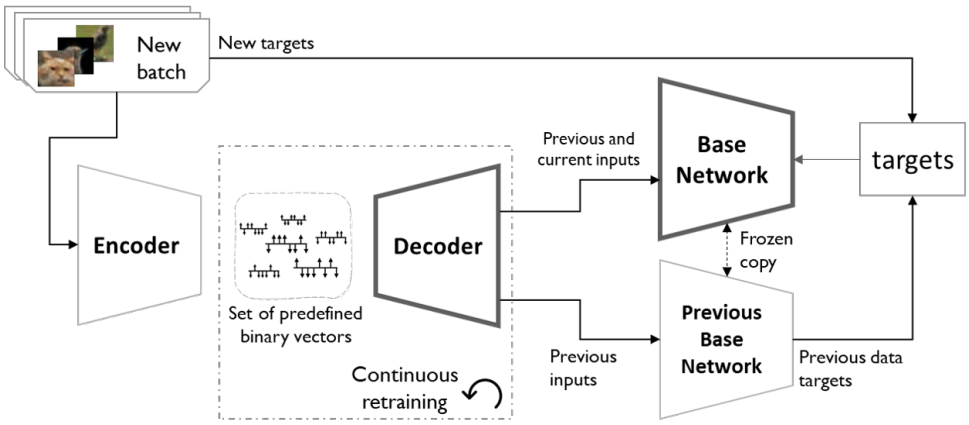


Figure 8.3.2. The BinPlay architecture consists of two feed-forward networks. The first one – **binary latent autoencoder** – finds the mapping of new training examples to the predefined binary codes and decodes them to the original training examples with the **decoder**. While training proper reconstruction of the binary codes into new images, we also feed the decoder with the combinations of previously seen codes and their reconstructions in the continuous retraining. We train the **base network** with the generated examples from the previous data and the current data samples. For the generated examples, we calculate appropriate labels with a frozen copy of the base network. Only the decoder and base network are needed to be stored in the memory buffer, while the predefined binary codes for past data are computed on the fly.

Once we define the binary codes for a given task, we need a procedure to map the original data into this set of codes. For a decoder to properly rehearse past samples, we need the mapping to provide a single binary code in the latent space for each data sample. We, therefore, propose the following greedy binary code assignment algorithm.

Algorithm 2 Designating regularisation losses for binary codes assignment in BinPlay.

$\mathbf{c}$ = set of binary codes, $\mathbf{x}$ = set of images, q = encoder, ϕ = encoders weights
put elements of X in a random order
for all x_i from $\mathbf{x}$ **do**
  $z_i = q_\phi(x_i)$
  c_i = the element of C closest to z_i
  $\mathbf{c} = \mathbf{c} \setminus \{c_i\}$
  $L_i(z) = \|z - c_i\|^2$
end for

For the set of images $\mathbf{x}$, we generate a set of binary codes $\mathbf{c}$, such that $|\mathbf{x}| = |\mathbf{c}| \ll 2^n$, where n is a size of the latent space. Then, while training our binary latent autoencoder, we extend the original reconstruction loss with an additional regularization term. To compute this term, we assign a yet unassigned binary code to each sample that lies at the lowest distance to the sample. Then, we calculate the distance between data embedding in the latent space and its associated binary code as a L_2 norm. The final regularization is a sum of the distances computed across all of the images from the task. Since this procedure is highly affected by the ordering of the samples, in each epoch, we shuffle the input images $\mathbf{x}$. Alg. 2 presents a general overview of the assignment procedure.

8.3.4. Training

With the building blocks of BinPlay introduced above, we can now describe how to train the whole model. Fig. 8.3.2 overviews the training procedure.

On the arrival of a new task, we first train the binary latent autoencoder. We train its encoder to assign new examples to their binary codes and the decoder to reconstruct them back to the original form. To retain the knowledge derived from the previous data, we use the training procedure as defined in Sec. 8.3.1. To enable faster convergence, we start training autoencoder without enforcing the binary regularisation of the latent space and turn the binary code assignment of Alg. 2 after a couple of *warm-up* rounds. This allows the autoencoder to cluster similar data samples in the latent space before mapping them to the allowed binary codes. Once

the encoder converges on the mappings between the input points and their binary codes, we save the mappings and omit the encoding loss to focus on the decoder.For training simplicity, we use the binary code values of {-1,1}, instead of {0,1}, as this facilitates the decoding procedure.

Our base network aims to minimise the loss averaged over current data and a sample of the previous tasks. On the arrival of a new task, we train the network with the new samples together with the samples regenerated from the binary codes sampled uniformly from the allocated pool in the autoencoder latent space. The new samples are processed through the autoencoder before using them as an input for training so that the base network does not focus on differences between the real and generated data but rather on the general image features that enable accurate classification. For the regenerated past samples, we train the base network to predict the same output distribution values as its copy – frozen before the training. Consequently, for classification tasks, we use soft targets Hinton et al. (2015) instead of the ground truth labels. Tab. 8.3.1 shows the results of the ablation study supporting the design choices presented in this section.

Modification	Results
Reference configuration	23.8
+ processing new samples through autoencoder	54.0
+ soft targets	63.3

Table 8.3.1. Ablation study for the training procedures of the base network. We report the average classification accuracy after the last task on the CIFAR-10 dataset. As a reference configuration, we take a base network trained on both original (current task) and generated (previous tasks) data samples. Preprocessing incoming new samples with an autoencoder and using soft targets improves the performance of our final continual learning model.

8.4. Experimental Study

We evaluate BinPlay on three commonly used benchmarks: MNIST LeCun et al. (2010), Fashion-MNIST Xiao et al. (2017), and CIFAR-10 Krizhevsky et al. (2009). To simulate the real setting of continual learning, we evaluate our model with the *class-incremental* scenario. This means that in each task, we introduce new classes to the scope of our model. In particular, we follow the split-MNIST split--FashionMNIST and split-CIFAR procedure that divides the dataset into 5 separate tasks with all training examples of classes: {0,1}, {2,3}, {4,5}, {6,7}, {8,9}, accordingly.

MNIST and Fashion-MNIST

For both MNIST datasets, we propose the architecture based on a convolutional neural network. For the encoder and the decoder, we use 3 convolutional/transposed convolutional layers and 2 fully connected layers around the latent space of size 200.

Our classifier is based on the LeNet Lecun et al. (1998) architecture with 3 convolutional layers followed by two fully connected ones. We use batch normalisation and dropout. Detailed implementation information can be found in the released codebase.

CIFAR-10 For the CIFAR-10 dataset, we employ similar network architectures as for MNIST and Fashion-MNIST but with a greater number of filters. Additionally, to simplify the encoding procedure for more complex images, we extend the binary latent size to 1000. For the classifier, we only adjust the previous LeNet architecture to the CIFAR-10 image size, which is $32 \times 32 \times 3$ pixels.

Data storage	Model	MNIST	Fashion MNIST	CIFAR–10
Memory buffer	GEM Lopez-Paz and Ranzato (2017b)	91.8 ± 0.3	70.3 ± 0.7	17.5 ± 1.6
	iCARL Rebuffi et al. (2017)	71.7 ± 0.5	67.7 ± 0.4	32.4 ± 2.1
	ER Chaudhry et al. (2019)	84.5 ± 1.6	70.2 ± 1.7	41.3 ± 1.9
	ER–MIR Aljundi et al. (2019)	91.8 ± 0.5	69.7 ± 2.7	47.6 ± 1.1
Both	AQM Caccia et al. (2020)	93.6 ± 0.7	67.4 ± 0.3	51.4 ± 2.2
Generative replay	GEN-MIR Aljundi et al. (2019)	86.6 ± 0.3	52.4 ± 1.5	18.8 ± 0.9
	OCDVAE Mundt et al. (2020b)	93.2 ± 3.7	69.9 ± 1.7	21.6
	GR Shin et al. (2017)	92.5 ± 0.5	68.0 ± 0.9	27.3 ± 1.3
	GR+distill Van de Ven and Tolias (2019)	95.6 ± 0.2	78.1 ± 1.0	28.4 ± 0.3
	RTF van de Ven and Tolias (2018)	95.1 ± 0.3	75.2 ± 0.8	28.7 ± 0.2
	BinPlay (ours)	**97.2 ± 0.6**	**81.4 ± 0.9**	**63.3 ± 1.4**

Table 8.4.1. Average accuracy after the final task in the class incremental scenario (in % ± SEM). We report the results for memory-buffer based (top) and generative replay (bottom) methods on MNIST, Fashion MNIST, and CIFAR-10. Our approach clearly outperforms competitive approaches on all three benchmarks.

8.4.1. Results

We compare the accuracy of our models' predictions. For that purpose, we run the experiment through all 5 tasks to report the average accuracy on the whole test-set after the final one. For the related methods, we report the results from the original works. However, for several techniques, authors openly state that their methods do not scale well for more complex datasets such as CIFAR-10 Aljundi et al. (2019); Lesort et al. (2019). Hence, in case of missing evaluations on any benchmarks, we ran the experiments with the code provided by the authors and

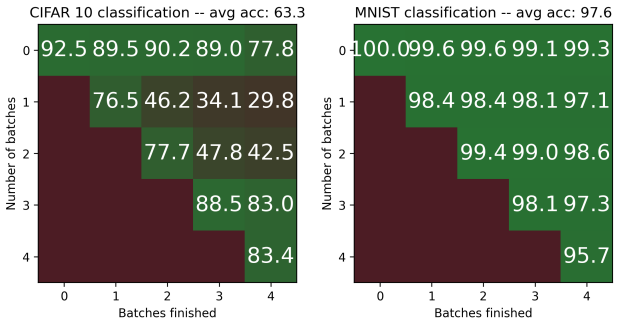
report the results averaged across 3 runs. As presented in Tab. 8.4.1, those experiments demonstrate that our solution clearly outperforms other generative replay approaches on all benchmark datasets. Additionally, in Fig. 8.4.2 we show that our BinPlay is only slightly affected by the number of consecutive tasks, and the deterioration of the results is rather the effect of the increasing complexity of the problem.

This is mainly thanks to the fact that our model can recreate sharp images with high-quality visual features. These are a valuable source of previous examples for the base classifier. To support this claim in Fig. 8.4.1, we show that our method is able to recreate rich images even after four tasks of not related data examples. This is contrary to the other generative rehearsal procedures such as RtF van de Ven and Tolias (2018) that produce blurry images already from the start.

Figure 8.4.1. Examples of the reconstructions (BinPlay) and generations (RtF van de Ven and Tolias (2018)) of the images from the CIFAR-10 dataset. Our method is able to recreate high-quality images even after four following tasks. On the contrary to the current state-of-the-art generative replay method (RtF), images are sharp and allow the classifier to rehearse also high-frequency features from regenerated images.

We also measure the memory footprint of our generative model. We compare it with the current generative rehearsal methods and the space requirements for the memory-buffer based ones. The results of this comparison are shown in Tab. 8.4.2. Our model is much more compact than the other generative rehearsal ones. That is thanks to the fact that we have to store only the convolutional decoder of the two parts of the generative autoencoder.

When compared to the rehearsal models with the buffer, our solution requires more space than experience replay methods. However, space requirement for our BinPlay architecture is constant. Our model does not grow with the number of tasks served, contrary to the buffer of standard rehearsal methods.



(a) **(b)**

Figure 8.4.2. Visualisation of the classification accuracy observed on the consecutive tasks. For CIFAR-10 (left), the accuracy of our model deteriorates steadily, and it is related to the increasing complexity of the general classification problem rather than the task index. For simple datasets, such as MNIST (right), we can observe steady performance through all tasks.

8.4.2. Future work

The BinPlay approach for generative rehearsal presented in this work relies on a binary latent autoencoder and the construction of binary codes living in the latent space. Although the design choices we describe already lead to a significant performance improvement of BinPlay over the competing methods, we can envision several new research paths that are enabled by our work. The currently used binary encoding parameterisation presented in Sec. 8.3 relies on a simple yet effective method to populate latent space. One potential direction of future research is the incorporation of other parameterisation methods, including trainable models with divergence loss functions. Another path of future work can explore other than binary latent space types and the corresponding encodings. In the future work, we also plan to discuss the vulnerability of our method to the diverse types of concept drifts Losing et al. (2017), defined as the observable phenomena of changing data distributions between tasks. Finally, we believe that the work presented here is a stepping stone towards discovering the relationship between generative models used in rehearsal and other compression techniques. Are generative models indeed used to compress previously learned knowledge or rather to represent the pattern for generating images of the same kind? Our empirical evaluation indicates that it is rather the latter, yet further research can definitely shed more light on this question.

Model	MNIST	Fashion MNIST	CIFAR–10
Memory-buffer based methods	0.7	0.7	2.9
AQM Caccia et al. (2020)	1.3	1.3	2.4
OCDVAE Mundt et al. (2020b)	115.9	115.9	-
GEN-MIR Aljundi et al. (2019)	5.0	11.8	34.0
GR Shin et al. (2017)	4.4	15.6	32.3
GR+distill Van de Ven and Tolias (2019)	4.4	15.6	32.3
RTF van de Ven and Tolias (2018)	4.4	15.6	32.3
BinPlay (ours)	4.6	4.6	21.0

Table 8.4.2. Memory requirements of continual learning method (in MB). For generative replay methods, it is calculated as the memory required to store the generative model weights. For memory-buffer based methods, it is the size of the buffer of 100 memories after the last task. We do not take into account any compression mechanisms neither for images nor for models. However, AQM Caccia et al. (2020) method includes an embedded compression mechanism, and therefore the corresponding results are presented here for the sake of completeness. For generative replay methods (bottom), BinPlay requires comparable (MNIST) or smaller (Fashion MNIST and CIFAR-10) amount of memory comparing to the competing models while yielding higher accuracy, as shown in Tab. 8.4.1. Moreover, the bigger and more complex the dataset, the more profound the memory savings offered by BinPlay are.

8.5. Conclusions

In this work, we proposed a novel approach for continual learning with generative replay. Our BinPlay approach introduces a novel binary latent space autoencoder architecture to embed past tasks as binary codes that can be later reconstructed on the fly from a simple and deterministic parameterisation function. Inspired by memory-buffer based models, we used binary latent space to store past data inputs as efficient binary codes while avoiding typical shortcomings of competing generative models that regenerated similar but lower-quality samples from the past. The evaluation comparison of our approach against state-of-the-art models clearly shows the superiority of BinPlay in terms of average accuracy on three benchmark datasets at a similar or smaller memory footprint than the competing generative rehearsal methods.

9. Multiband VAE: Latent Space Alignment for Knowledge Consolidation in Continual Learning

Title	Multiband VAE: Latent Space Alignment for Knowledge Consolidation in Continual Learning
Authors	Kamil Deja, Paweł Wawrzyński, Wojciech Masarczyk, Daniel Marczak, and Tomasz Trzciński
Journal	Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence IJCAI2022
Year	2022
DOI	10.24963/ijcai.2022/402

Preface

In this chapter, we extend the previous work and place a generative model in the focal point of the continually trained system. We postulate that this family of methods is a promising direction for a general system of continuous knowledge consolidation. We base this statement on three fundamental aspects of generative models.

Firstly, in our previous work (Masarczyk et al., 2021), we showed that neural networks trained with a reconstructive task are less prone to the problem of catastrophic forgetting. In this work, we analysed how data representations of the model change when retrained with a new task, depending on the training objective. Our experiments indicated that forgetting in generative models is gradual and monotonic, contrary to the discriminative models, where it usually happens abruptly. Moreover, we show that generative models learn more transferable features better aligned between tasks.

Secondly, the ability of generative models to generate synthetic samples provides a complementary method for generative rehearsal. This is contrary to the discriminative tasks, where an additional structure in the form of a memory buffer or additional generative model has to be used to store past data examples. Moreover, rehearsal samples generated by a generative model align well with how the model encodes the information.

Finally, as we present in this chapter, low-dimensional data representations of the latent generative models are more efficient for continual knowledge consolidation. This is primarily thanks to the well-structured knowledge encoded by the generative autoencoder. In particular, in this chapter, we introduce Multiband VAE – a method for continual alignment of knowledge represented in the variational autoencoder's latent space. We propose a new approach, where an update of the continually trained VAE is divided into two parts. In the first one, we use a fresh model to learn the encoding of newly available data. In the second one, we align previous and new data with an additional MLP projector working in the latent space of the VAE.

In a series of experiments, we show that our approach can properly encode data from entirely different distributions as well as partially similar examples. In the latter situation, where we retrain the model with a new portion of partially similar data, we show that we can improve the model's performance, showing positive backward and forward knowledge transfer.

Abstract

We propose a new method for unsupervised generative continual learning through realignment of the Variational Autoencoder's latent space. Deep generative models suffer from *catastrophic forgetting* in the same way as other neural structures. Recent *generative continual learning* works approach this problem and try to learn from new data without forgetting previous knowledge. However, those methods usually focus on artificial scenarios where examples share almost no similarity between subsequent portions of data – an assumption not realistic in the real-life applications of continual learning. In this work, we identify this limitation and posit the goal of generative continual learning as a knowledge accumulation task. We solve it by continuously aligning latent representations of new data that we call *bands* in additional latent space where examples are encoded independently of their source task. In addition, we introduce a method for controlled forgetting of past data that simplifies this process. On top of the standard continual learning benchmarks, we propose a challenging knowledge consolidation scenario and show that the proposed approach outperforms state-of-the-art by up to twofold across all experiments and the additional real-life evaluation. To our knowledge, Multiband VAE is the first method to show forward and backward knowledge transfer in generative continual learning.

9.1. Introduction

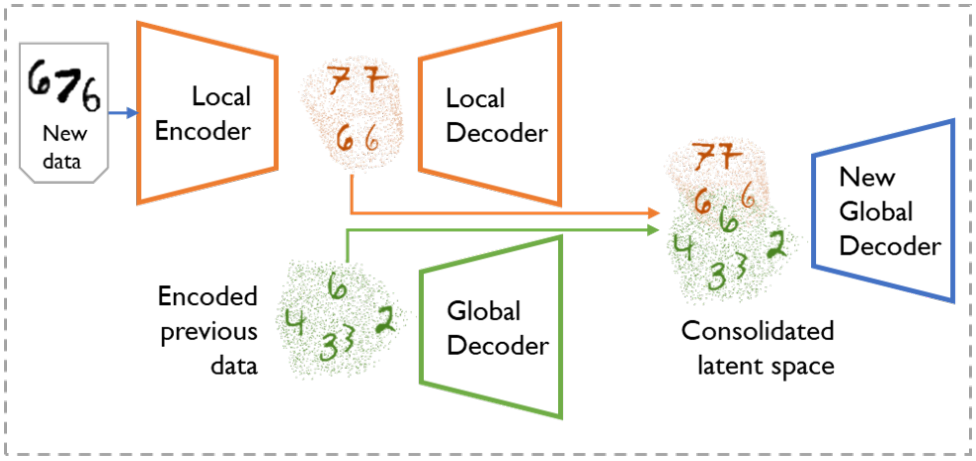


Figure 9.1.1. Overview of our Multiband VAE. With each new task, we first learn a *local* copy of our model to encode new data examples. Then we consolidate those with our current global decoder - main model which is able to generate examples from all tasks.

Recent advances in generative models (Goodfellow et al., 2014b; Kingma and

Welling, 2014) led to their unprecedented proliferation across many real-life applications. This includes high energy physics experiments at Large Hadron Collider (LHC) at CERN, where they are employed to speed up the process of particles collisions simulations (Paganini et al., 2018; Deja et al., 2020; Kansal et al., 2021).

Those applications are possible, thanks to the main objective of generative methods, which is the modelling of complex data manifolds with simpler distributions. Unfortunately, this goal remains difficult to deliver in real-life situations where training data is presented to the model in separate portions, e.g., from consecutive periods of data gathering at CERN. The distributions of data within these portions often vary significantly, hence updating model with new examples leads to *catastrophic forgetting* of previous knowledge. In generative modelling this is observed through limited distribution of generated examples.

Generative continual learning methods aim to address these challenges usually in one of three ways: through regularisation (e.g. (Nguyen et al., 2018)), adjustment of the structure of a network to the next task (e.g. (Rao et al., 2019)), or rehearsal of previously seen samples when training with new data (e.g. (Rebuffi et al., 2017)). Nevertheless, practical applications of those methods are yet limited, so most of them focus on the artificial class-incremental (CI) training scenario. In this approach, consecutive portions of data (tasks) contain disjoint classes and share almost no similarity. While this is the most difficult scenario for discriminative models, we argue that the assumption of classes separation greatly simplifies the problem in generative modelling where task index might be used without reducing the method's generality (detailed discussion in the appendix).

Moreover, the assumption of task independence in CI scenario reduces the complexity of continual learning (Ke et al., 2021). Therefore, in this work, we postulate to investigate the adaptation of generative continual learning methods to the ever-changing data distribution. While, for the CI scenario, we expect no forgetting of previous knowledge, in other scenarios, where model is retrained with additional partially similar data, we should aim for performance improvement. This can be observed through *forward knowledge transfer* – higher performance on a new task, thanks to already incorporated knowledge, and *backward knowledge transfer* – better generations from previous tasks, when retrained on additional similar examples (Lopez-Paz and Ranzato, 2017a).

Therefore, to simulate real-life conditions, we prepare a set of diversified continual learning scenarios with data splits following Dirichlet distribution, inspired by a similar approach in federated learning (Hsu et al., 2019). Our experiments indicate that this is indeed a more challenging setup for the majority of recent

state-of-the-art continual generative models, which lack sufficient knowledge sharing between tasks.

To mitigate this problem, we propose a Multiband VAE. The core idea behind our method is to split the process of model retraining into two steps: (1) a local encoding of data from the new task into a new model's latent space and (2) a global rearrangement and consolidation of new and previous data. In particular, we propose to align local data representations from consecutive tasks through the additional neural network. In reference to the way how radio spectrum frequencies are allocated, we name data representations from different tasks *bands*. As in telecommunication, our goal is to limit interference between bands. However, we train our model to align parts that represent the same or similar data. To support knowledge consolidation between different bands, we additionally propose a controlled forgetting mechanism that enables the substitution of degraded reconstructions of past samples with new data from the current task.

The main contributions of this work are:

- A novel method for generative continual learning of Variational Autoencoder that counteracts catastrophic forgetting while being able to align even partially similar tasks at the same time.
- A simple method for controlled forgetting of past examples whenever a new similar data is presented.
- A novel knowledge consolidation training scenario that underlines limitations of recent state-of-the-art methods.

9.2. Related Works

In this work, we directly extend the basic idea of training generative models in continual learning described as Generative Replay (Lesort et al., 2019). We also incorporate the disentanglement method with binary latent space, similarly to Ramapuram et al. (2020) In the experimental setup we compare our solution with different methods for continual learning of generative models such as Generative Replay (Lesort et al., 2019), Variational Continual Learning (Nguyen et al., 2018), HyperCL (von Oswald et al., 2019), CURL (Rao et al., 2019), Lifelong VAE (Ramapuram et al., 2020) and Lifelong-VAEGAN (Ye and Bors, 2020) overviewed in Sec. 7.2.

9.3. Method

In this section, we introduce Multiband VAE – a method for consolidating knowledge in a continually learned generative model. We propose to split generative replay training into two parts: (1) a local training that allows us to build a new data representations band in the latent space of VAE, and (2) global training where we attach a newly trained band to the already trained global model. As a part of the global training, we propose a controlled forgetting mechanism where we replace selected reconstructions from previous tasks with currently available data.

9.3.1. Knowledge Acquisition – Local Training

In the local training, we learn a new data representations band by training a VAE using only currently available data.

Let $\mathbf{x}^i_j$ denote the j-th sample of i-th task. Then, for given sample $\mathbf{x}^i_j$, and latent variable λ^i_j we use a decoder p_θ, which is trained to maximise posterior probability $p(\mathbf{x}^i_j|\lambda^i_j)$. To get the latent variable λ^i_j, we use encoder q_ϕ parameterised with weights vector ϕ that approximates probability $q(\lambda^i_j|\mathbf{x}^i_j)$.

To simplify the notation, let us focus on specific task i and drop the index. As in standard VAE, we follow optimisation introduced by Kingma and Welling (2014) that maximises the variational lower bound of log likelihood:

$$\mathcal{L}_{local} = \max_{\phi,\theta} \mathbb{E}_{q(\lambda|\mathbf{x})}[\log p(\mathbf{x}|\lambda)] - D_{KL}(q(\lambda|\mathbf{x})\|\mathcal{N}(\vec{0}, I))). \tag{9.1}$$

where ϕ and θ are weights of encoder and decoder respectively. In the first task, this is the only part of the training, after which local decoder is remembered as a global one. In other cases we drop local decoder.

9.3.2. Shared Knowledge Consolidation

In the second – global part of the training, we align the newly trained band with already encoded knowledge. The simplest method to circumvent interference between bands is to partition the latent space of VAE and place new data representation in a separate area of latent space. However, such an approach limits information sharing across separate tasks and hinders forward and backward knowledge transfer. Therefore, in Multiband VAE we propose to align different latent spaces through an additional neural network that we call *translator*. Translator maps individual latent spaces which are conditioned with task id into the common global

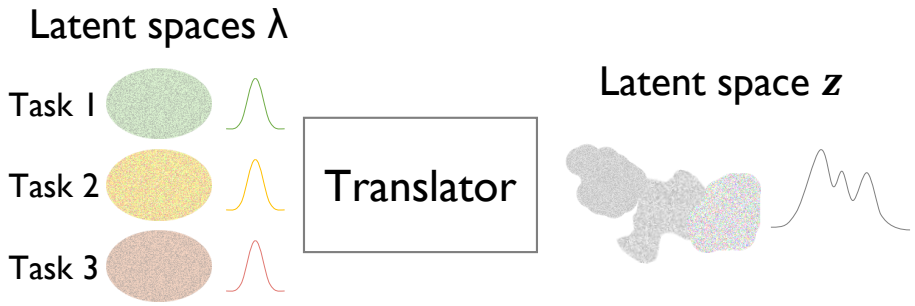


Figure 9.3.1. Our translator maps individual regularised latent spaces λ created by different local models to one global latent space $\mathbf{z}$, where examples are stored independently of their source task.

one where examples are stored independently of their source task, as presented in Fig 9.3.1.

To that end, we define a translator network $t_\rho(\lambda^i, i)$ that learns a common alignment of separate latent spaces λ^i conditioned with task id i to a single latent variable $\mathbf{z}$, where all examples are represented independently of their source task. Finally, we propose a global decoder $p_\theta(\mathbf{x}|\mathbf{z})$ that based on distribution approximated with latent variables $\mathbf{z}$ learns to approximate original data distribution $\mathbf{x}$.

To counteract forgetting, when training translator and global decoder we use auto-rehearsal as in standard generative replay, with a copy of the translator and decoder frozen at the beginning of the task. As training pairs, we use combination of original images $\mathbf{x}$ with their encodings from local encoder λ, and for previous tasks, random values λ with generations $\mathbf{x}$ reconstructed with a frozen translator and global decoder. Fig. 9.3.2 presents the overview of this procedure.

We start translator training with a frozen global decoder, to find the best fitting part of latent space $\mathbf{z}$ for a new band of data without disturbing previous generations. For that end we minimise the reconstruction loss:

$$\mathcal{L}_{translator} = \min_{\rho} \sum_{i=1}^{k} ||\mathbf{x}^i - p_\theta(t_\rho(\lambda^i, i))||_2^2, \tag{9.2}$$

where k is the number of all tasks.

Then, we optimise parameters of translator and global decoder jointly, minimising the reconstruction error between outputs from the global decoder and training examples

$$\mathcal{L}_{global} = \min_{\rho,\theta} \sum_{i=1}^{k} ||\mathbf{x}^i - p_\theta(t_\rho(\lambda^i, i))||_2^2, \tag{9.3}$$

To generate new example t with Multiband VAE, we randomly sample task id $i \sim$

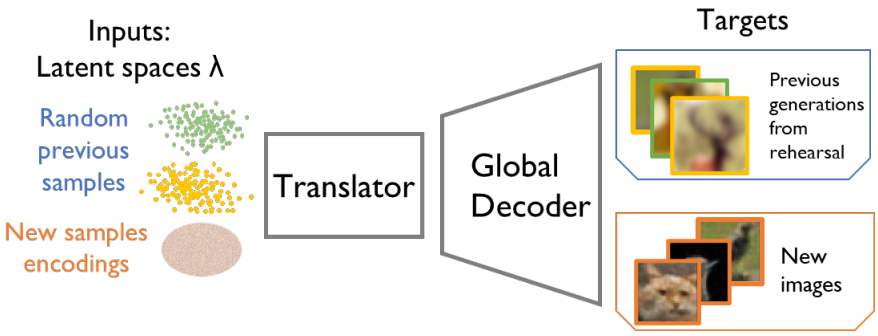


Figure 9.3.2. We train our translator and global decoder with new data encoded to latent space λ associated with original images, and samples of previous data generations generated in a standard rehearsal schema.

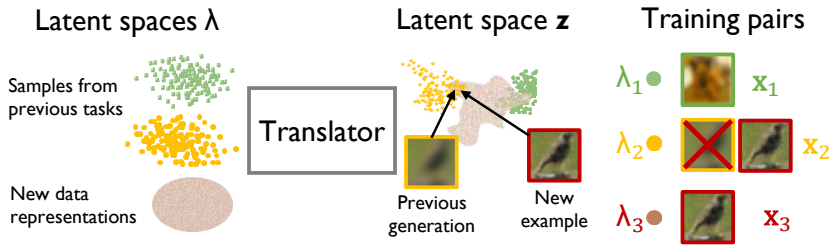


Figure 9.3.3. When creating rehearsal training pairs with generations from previous data examples, we calculate the similarity between sampled example and the closest currently available data sample in the common latent space $\mathbf{z}$. If this similarity is above a given threshold, we allow forgetting of the previous reconstruction by substituting the target generation with a currently available similar image.

$\mathcal{U}(\{1,\ldots,k\})$, where k is the number of all tasks and latent representation $\lambda_t \sim \mathcal{N}(\vec{0}, I)$. These values are mapped with translator network to latent variable $\mathbf{z}_t$, which is the input to global decoder to generate $\mathbf{x}_t$. Therefore, translator and global decoder are the only models that are stored in-between tasks.

9.3.3. Controlled Forgetting

In a real-life scenario, it is common to encounter similar data examples in many tasks. In such a case, we would like our continuously trained model to refresh the memory of examples instead of combining vague, distorted memories with new instances. Therefore, we propose a mechanism for controlled forgetting of past reconstructions during the translator and global decoder joint training. To that end, when creating new training pairs, we compare representations of previous data reconstructions generated as new targets with representations of data samples from the current task in the common latent space $\mathbf{z}$. If these representations are similar

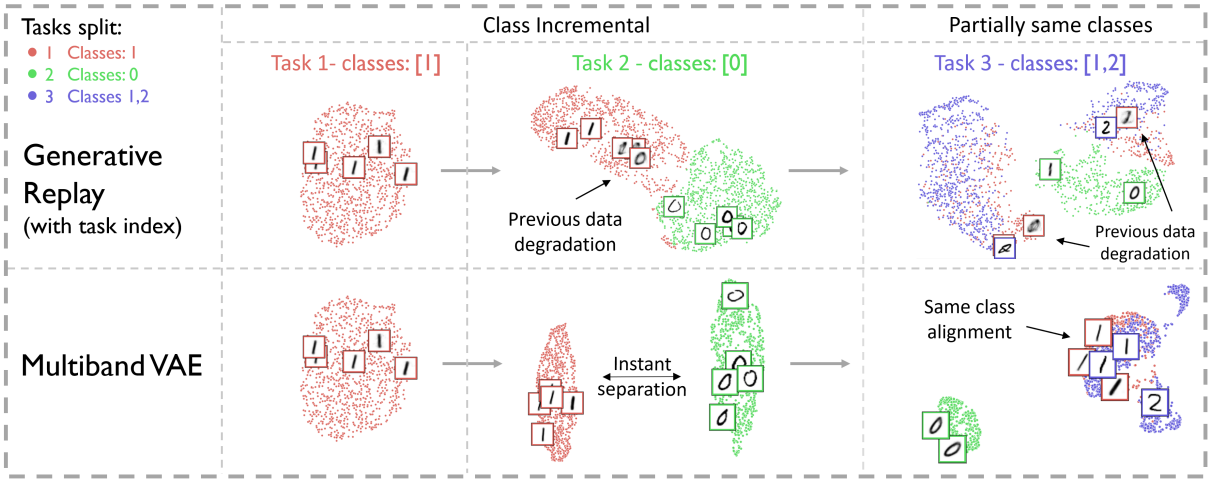


Figure 9.3.4. Visualisation of latent space $\mathbf{z}$ and generations from VAE in standard Generative Replay and our multiband training for the three tasks (different colours) in a case of entirely different new data distribution, and partially same classes. GR does not instantly separate data from different tasks, which results in the deformation of previously encoded examples. Contrary, our Multiband VAE can separate representations from different classes while properly aligning examples from the same new class if present.

enough, we substitute previous data reconstruction with the current data sample as presented in Fig. 9.3.3.

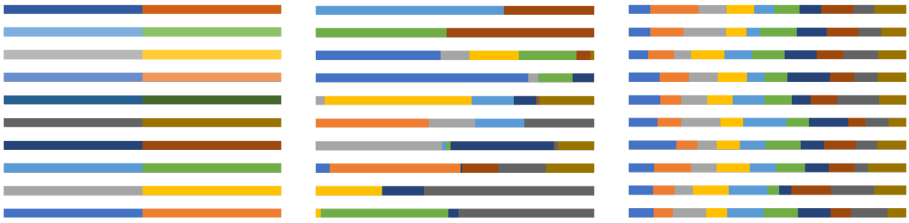
More specifically, when training on task i, we first create a subset $\mathcal{Z}^i = t_\rho(q_\phi(\mathbf{x}^i), i)$ with representations of all currently available data in joint latent space $\mathbf{z}$. Now, for each data sample $\mathbf{x}^l_j$ generated as a rehearsal target from previous task $l < i$ and random variable λ^l_j, we compare its latent representation $z_j = t_\rho(\lambda^l_j, j)$ with all elements of set $\mathcal{Z}^i$

$$sim(z_j) := \max_{z_q \in \mathcal{Z}^i} cos(z_j, z_q). \tag{9.4}$$

If $sim(z_j) \geq \gamma$ we substitute target sampled reconstruction $\mathbf{x}^l_j$ with respective original image from $\mathbf{x}^i$. Intuitively, γ controls how much do we want to forget from task to task, with $\gamma = 0.9$ being a default value for which we observe a stable performance across all benchmarks.

9.4. Experiments

To visualise the difference between Generative Replay and Multiband VAE, in Fig. 9.3.4 we present a toy-example with the MNIST dataset limited to 3 tasks with data examples from 3 classes. When presented with data from a new distribution (different class in task 2), our method places a new band of data in a separate part of a common latent space $\mathbf{z}$. On the other hand, the standard generative replay model



Class incremental Dirichlet α=1 Dirichlet α=100

Figure 9.4.1. Class splits for different continual learning scenarios. In class incremental split each task consists of separate classes. For $\alpha = 1$ Dirichlet distribution, we have highly imbalanced splits with randomly occurring dominance of one or two classes. For higher values of parameter α, classes are split almost equally.

learns to transform some of the previous data examples into currently available samples before it can distinguish them, even with additional conditioning on task identity. At the same time, when presented data with partially same classes as in task 3, our translator is able to properly align bands of data representations so that similar data examples (in this case ones) are located in the same area of latent space $\mathbf{z}$ independently of the source task, without interfering with zeros and twos.

9.4.1. Evaluation Setup

For fair comparison, in all evaluated methods we use a Variational Autoencoder architecture similar to the one introduced by Nguyen et al. (2018), with nine dense layers. However, our Multiband VAE is not restricted to any particular architecture, so we also include experiments with a convolutional version. The exact architecture and training hyperparameters are enlisted in the appendix and code repository[1]. We do not condition our generative model with class identity since it greatly simplifies the problem of knowledge consolidation and applies to all evaluated methods. However, similarly to Ramapuram et al. (2020), we use additional binary latent space trained with Gumbel softmax (Jang et al., 2016).

9.4.2. Evaluation

To assess the quality of our method, we conduct a series of experiments on benchmarks commonly used in continual learning (MNIST, Omniglot (Lake et al., 2015)) and generative modelling – FashionMNIST (Xiao et al., 2017). Since the performance of VAE on diverse datasets like CIFAR is limited, in order to evaluate how our method scales to more complex data, we include tests on CelebA (Liu et al., 2015b). For each dataset, we prepare a set of training scenarios designed to eval-

[1] `https://github.com/KamilDeja/multiband_vae`

Num. tasks	Split-MNIST Class Incremental 5			MNIST Dirichlet $\alpha = 1$ 10			Split-Fashion MNIST Class Incremental 5			Fashion MNIST Dirichlet $\alpha = 1$ 10			CERN Class Inc. 5
Measure	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑	FID ↓	Prec ↑	Rec ↑	Wass ↓
SI	129	77	80	153	75	76	134	28	24	140	21	19	21.1
EWC	136	73	82	120	79	83	126	25	25	137	24	22	29.7
Generative replay	120	79	87	254	70	65	96	43	58	133	35	43	11.1
VCL	68	85	94	127	78	80	104	30	32	138	21	20	24.3
HyperCL	62	91	87	148	78	75	108	46	33	155	35	21	7.8
CURL	107	95	77	181	84	74	86	47	64	83	46	56	16.8
Livelong-VAE	173	75	72	224	63	73	131	33	62	201	9	49	7.7
Livelong-VAEGAN	48	**98**	89	131	90	83	78	54	**79**	108	54	64	15.1
Multiband VAE	**24**	94	**97**	**41**	**92**	**96**	**61**	**66**	69	**82**	**62**	**65**	**6.6**
Multiband VAE (conv)	23	92	98	30	92	97	56	65	72	77	58	69	8.1

Table 9.4.1. Average FID and distribution Precision (Prec) and Recall (Rec) or Wasserstein distance between original and generated simulation channels, after the final task in different data incremental scenarios. Our method with vanilla architecture outperforms competing solution.

Num. tasks	Split-Omniglot Class Incremental 5			Split-Omniglot Class Incremental 20			Omniglot Dirichlet $\alpha = 1$ 20			FashionM→MNIST Class Incremental 10			MNIST→FashionM Class Incremental 10		
Measure	FID↓	Prec↑	Rec↑	FID↓	Prec↑	Rec↑	FID↓	Prec↑	Rec↑	FID↓	Prec↑	Rec↑	FID↓	Prec↑	Rec↑
SI	48	87	81	115	64	28	140	18	16	146	18	15	157	21	19
EWC	46	88	81	106	68	31	106	74	38	119	72	30	133	25	23
Generative replay	45	88	82	74	72	62	92	75	53	99	36	45	111	24	39
VCL	48	87	82	122	62	21	127	71	25	81	45	51	79	45	55
HyperCL	54	86	76	98	86	45	115	84	38	128	31	28	143	30	28
CURL	22	95	**95**	**31**	**96**	**92**	**26**	94	**92**	98	**69**	42	122	47	37
Lifelong-VAE	49	87	83	79	83	59	93	83	51	173	13	50	200	12	52
Lifelong-VAEGAN	31	96	90	71	83	70	63	85	78	127	34	61	91	52	**73**
Multiband VAE	**21**	**97**	93	33	95	86	41	**95**	83	**51**	65	**70**	**49**	**67**	**73**
Multiband VAE (conv)	12	98	96	24	95	91	24	96	91	49	68	70	49	70	70

Table 9.4.2. Average Fréchet Inception Distance (FID) and distribution Precision (Prec) and Recall (Rec) after the final task in different data incremental scenarios. In more challenging datasets Multiband VAE outperforms competing solutions.

uate various aspects of continual learning. This is the only time we access data classes, since our solution is fully unsupervised.

To assess whether the model suffers from catastrophic forgetting, we run class incremental scenarios introduced by Van de Ven and Tolias (2019). However, CI simplifies the problem of learning data distribution in the generative model's latent space since the identity of the task conditions final generations. Therefore, we also introduce more complex data splits with no assumption of independent task distributions. To that end, we split examples from the same classes into tasks, according to the probability $q \sim Dir(\alpha p)$ sampled from the Dirichlet distribution, where p is a prior class distribution over all classes, and α is a *concentration* parameter that controls similarity of the tasks, as presented in Fig. 9.4.1. In particular, we exploit the Dirichlet $\alpha = 1$ scenario, where the model has to learn the differences between tasks while consolidating representations for already known classes. In such a scenario we expect forward and backward knowledge transfer between tasks.

CelebA split	Class Incremental			Dirichlet $\alpha = 1$			Dirichlet $\alpha = 100$			Single split		
Num. tasks	5			10			10			1		
Measure	FID↓	Prec↑	Rec↑	FID↓	Prec↑	Rec↑	FID↓	Prec↑	Rec↑	FID↓	Prec↑	Rec↑
Separate models	103	31	21	105	24.5	7.6	109	28.4	10.6			
Generative Replay	105	23.4	14.9	109	14.6	7.4	102	17.2	11.6	88	35	30
Multiband VAE	95	28.5	23.2	93	33	22	89	36.2	28			

Table 9.4.3. Average FID, distribution Precision, and Recall after the final task on the CelebA dataset. Our Multiband VAE consolidates knowledge from separate tasks even in the class incremental scenario, clearly outperforming other solutions. With more even splits our method converges to the upper bound which is a model trained with full data availability.

To measure the quality of generations from different methods, we use the Fréchet Inception Distance (FID) (Heusel et al., 2017). As proposed by Bińkowski et al. (2018), for simpler datasets, we calculate FID based on the LeNet classifier pre-trained on the whole target dataset. Additionally, we report the precision and recall of the distributions as proposed by Sajjadi et al. (2018). As authors indicate, those metrics disentangle FID score into two aspects: the quality of generated results (Precision) and their diversity (Recall).

For each experiment, we report the FID, Precision, and Recall averaged over the final scores for each task separately. For methods that do not condition generations on the task index (CuRL and LifelongVAE), we calculate measures in comparison to the whole test set. The results of our experiments are presented in Tab. 9.4.1 and Tab. 9.4.2, where we show scores averaged over three runs with different random seeds.

To compare different continual-learning generative methods in a real-life scenario we also use real data from detector responses in the LHC experiment. Calorimeter response simulation is one of the most profound applications of generative models where those techniques are already employed in practice (Paganini et al., 2018). In our studies, we use a dataset of real simulations from Zero Degree Calorimeter in the ALICE experiment at CERN introduced by Deja et al. (2020), where a model is to learn outputs of 44×44 resolution energy depositions in calorimeter. Following Deja et al. (2020), instead of using FID, for evaluation, we benefit from the nature of the data and compare the distribution of real and generated channels – the sum of selected pixels that well describe the physical properties of simulated output. We report the Wasserstein distance between original and generated channels distribution to measure generations' quality. We prepare a continual learning scenario for this dataset by splitting examples according to their input energy, simulating changing conditions in the collider. In practice, such split lead to continuous change in output shapes with partial overlapping between tasks – sim-

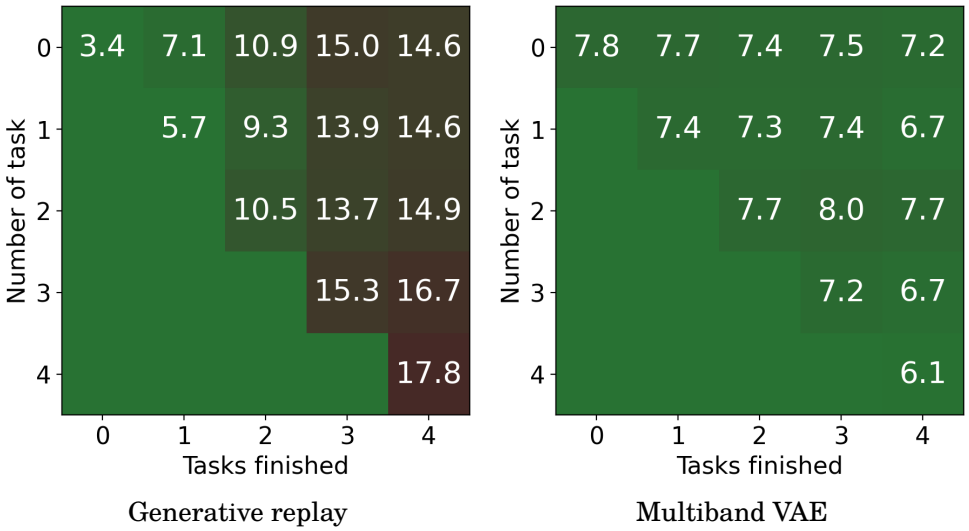


Figure 9.4.2. Comparison of Wasserstein distance ↓ between original simulation channels and generations from VAE trained with standard GR and our multiband training. Multiband VAE well consolidates knowledge with forward transfer (each row starts with better score) and backward knowledge transfer (improvement for some rows when retrained with more data). At the same time standard GR struggles to retain quality of generations on old tasks.

ilarly to what we can observe with Dirichlet based splits on standard benchmarks (see appendix for more details and visualisations).

As presented in Tab. 9.4.1, our model outperforms comparable methods in terms of quality of generated samples. Results of comparison on the Omniglot dataset with 20 splits (Tab. 9.4.2) indicate that for almost all of related methods, training with the data splits according to the Dirichlet $\alpha = 1$ distribution poses a greater challenge than the class incremental scenario. However, our Multiband VAE can precisely consolidate knowledge from such complex setups, while still preventing forgetting in CI scenario. This is only comparable to CURL that achieves this goal through additional model expansion. Experiments on more complex joint datasets, where examples are introduced from one dataset after another, indicate the superiority of Multiband VAE over similar approaches. In the real-life CERN scenario, our model also clearly outperforms other solutions. In Fig. 9.4.2 we present how generations quality for this dataset changes in standard generative replay and Multiband VAE, showing both forward and backward knowledge transfer in Multiband VAE.

Finally, we evaluate our model with a more complex dataset – CelebA with over 200 000 images of celebrity faces in 64x64 resolution. Based on annotated features, we split the dataset into 10 classes based on the hair colour/cover (blonde, black, brown, hat, bald or gray). In Tab. 9.4.3 we show the results of experiments with this dataset split in class incremental and Dirichlet scenarios. For class incremental scenario, Multiband VAE learns to separate bands of examples from different tasks

Modification	FID↓
Generative replay	254
+ Two step training	64
+ Translator	53
+ Binary latent space	44
+ Controlled forgetting	41
+ Convolutional model	30

Table 9.4.4. Ablation study on the MNIST dataset with Dirichlet $\alpha = 1$ distribution. Average FID after the last task.

with disjoint distributions, while results improve if in training scenario model is presented with more similar examples. In the latter case, with Dirichlet $\alpha = 100$ splits, our model reaches the quality of the upper bound, which is a standard Variational Autoencoder trained with full access to all examples in the stationary training.

Ablation study The main contribution of this work is a multiband training procedure, yet we also introduce several mechanisms that improve knowledge consolidation. Tab. 9.4.4 shows how those components contribute to the final score.

9.4.3. Memory Requirements and Complexity

The memory requirements of Multiband VAE are constant and equal to the size of the VAE with an additional translator, which is a small neural model with 2 fully connected layers. When training on the new task, our method requires additional temporary memory for the local model freed when finished. This is contrary to similar methods (HyperCL, VCL, CURL) which have additional constant or growing memory requirements. Computational complexity of our method is the same as for methods based on generative rehearsal (VCL, LifelongVAE, Lifelong-VAEGAN). In experiments, we use the same number of epochs for all methods, while for Multiband VAE we split them between local and global training.

9.5. Conclusion

In this work, we propose a new method for unsupervised continual learning of generative models. We observe that the currently employed class-incremental scenario simplifies the continual learning of generative models. Therefore, we propose a novel, more realistic scenario, with which we experimentally highlight the limitations of state-of-the-art methods. Finally, we introduce a new method for continual learning of generative models based on the constant consolidation of VAE's latent

space. To our knowledge, this is the first work that experimentally shows that with continually growing data with even partially similar distribution, we can observe both forward and backward performance improvement. Our experiments on various benchmarks and with real-life data show the superiority of Multiband VAE over related methods, with upper-bound performance in some training scenarios.

9.6. Appendix

In this supplementary material we present extended visualisations of the experiments with Multiband VAE, as well as the implementation details for our models. Finally, we show additional generations sampled from our generative model trained in the continual learning scenarios with the complex datasets such as combined MNIST → FashionMNIST and CelebA.

9.6.1. Discussion on the task index usage in generative continual learning

Access to the task number in continual learning of discriminative models simplifies the problem. It is mostly used when taking crucial decisions such as selecting the relevant part of the model for inference, or the final classification decision. In such cases, a need for task code greatly undermines the universality of a solution.

Contrary to the *discriminative* models, in *generative* case conditioning generation on task index does not influence or simplify the evaluation setting. The goal of a continually learned generative model is to generate an instance modelled on examples from *any* of the previous batches. Hence, to use a continually learned generative model in practice, we can randomly sample a task index (provided that it is lower than the total number of seen tasks) the same way we randomly sample input noise to the decoder or generator. In fact, training generative models with task index significantly simplifies a class incremental scenario, in which data distributions from separate tasks – with different classes are easily distinguishable from each other. In such case task index serves as an additional conditioning input imperceptibly leading to the conditional generative model. This limits the universality of proposed generative continual learning method.

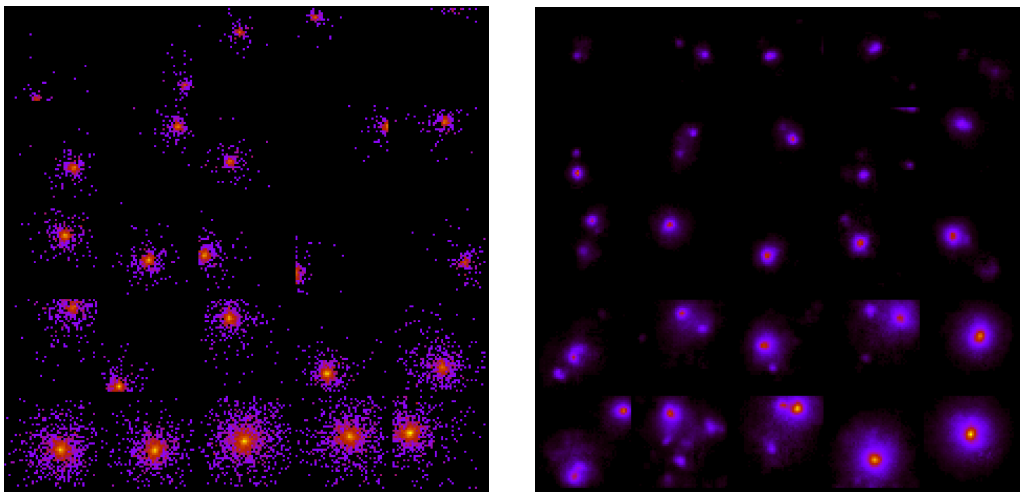
9.6.2. Models architectures

In this section, we describe in detail the architectures of VAE used in our experiments. The same models and hyperparameters can be found in the codebase [2].

Fully connected Variational Autoencoder For comparison with other methods we propose a simple VAE architecture with 9 fully connected layers which we use with simpler datasets: MNIST, Omniglot, FashionMNIST, CERN and combined datasets MNIST → FashionMNIST and FashionMNIST → MNIST.

In the encoder, we use three fully connected layers transforming input of 784 values through 512, 128 to 64 neurons. Afterwards, we map encoded images into continuous and binary latent spaces. For MNIST we use continuous latent space of

[2] `https://github.com/KamilDeja/multiband_vae`



(a) Original data **(b)** Generated simulations

Figure 9.6.1. Original simulations for Zero Degree Calorimeter responses and generations from our Multiband VAE trained in the class incremental scenario on CERN dataset (**in logarithmic scale**). We split original dataset into 5 tasks (each row of visualisation) with increasing energy of input particle. This results in continuously scaled size of the observed showers with partial overlapping between tasks. Multiband VAE well consolidates knowledge generating various outputs with full energy spectrum. Although because of the logarithmic scale generated examples seems blurred, this is of the low importance because of the extremely low values of darker/purple pixels.

size 8 and additional binary latent with size 4. For FashionMNIST and Omniglot we extend it to 12 continuous and 4 binary values.

The translator network takes three separate inputs: continuous encodings, binary encodings, and binary codes representing task number. We first process both binary inputs separately through two fully connected layers of 18 and 12 values for task codes and 8 and 12 neurons for binary encodings. Afterwards, we concatenate those three inputs: continuous noise from the encoder and two preprocessed binary encodings into a vector of size 32 for MNIST and 36 for Omniglot and FashionMNIST. We further process these values through two fully connected layers of 192 and 384 neurons which is the size of the second latent space $\mathbf{z}$

Our decoder consists of 3 fully connected layers with 512, 1024, and final 784 values. In each hidden layer of the model (except for the outputs of the encoder and translator) we use a LeakyRelu activation and sigmoid for the final one.

Convolutional VAE Our Multiband VAE is not restricted to any particular architecture, therefore we also include experiments with a convolutional version of our model. In this setup, for the encoder, we use 3 convolutional layers with 32, filers each of 4×4 kernel size and 2×2 stride. After that, we encode the resulting feature map of 288 features into the latent spaces of the same sizes as in the fully connected

model. For the translator network, we use a similar multilayered perceptron as in the fully connected model, however, we extend the dimensionality of latent space $\mathbf{z}$ to 512.

In the decoder, we use one fully connected layer that maps the output of the translator with 512 values into 2048 features. Those are propagated through 3 transposed convolution layers with 128, 64 and 32 filters of 4×4 kernel size and 2×2, 2×2, and 1×1 stride. The final transposed convolution layer translates filters into the final output with 4×4 kernel.

For the CelebA dataset, we extend our convolutional model. In the encoder, we use four convolutional layers with 50, 100, and 200 filters with 5×5 kernel size, followed by fully connected layer mapping 1800 features, through the layer of 200 neurons into the latent space of 32 neurons and binary latent space of 8 neurons. In the translator, we extend the fully connected combined layers into 800 and 1600 features which is a dimensionality of latent space Z. Our decoder decodes 1600 features from latent space through 3 transposed convolution layers with 400, 200, and 100 filters into the final output with 3 channels.

As in the fully connected model, we use LeakyReLU activations and additional batch normalization after each convolutional layer.

Training hyperparameters We train our models with the Adam optimiser, learning rate 0.001 and exponential scheduler with scheduler rate equal to 0.98. In our experiments, we train our model for 70 epochs of local training and 140 epochs of global training, with 5 epochs of shared knowledge discovery. We combine each mini-batch of original data examples with generations from previous tasks reaching up to:
mini_batch_size×num_tasks×0.5 samples per mini batch.

For the splits according to the Dirichlet distribution we substitute target generations with cosine similarity greater then 0.95. For class incremental scenario we set this parameter to 1. Nevertheless, our experiments indicate that lowering this value to 0.9 does not influence model's performance.

9.6.3. Real life CERN dataset

In this work we evaluate different continual learning generative methods with real-life example of particle collisions simulation dataset. For that end we use data introduced in Deja et al. (2020) that consists of 117 817 Zero Degree Calorimeter responses to colliding particles, calculated with the full GEANT4 Incerti et al. (2018) simulation tool. Each simulation starts with a single particle with a given properties (such as momenta, type or energy) propagated through the detector with

simulation tool that calculates interaction of a particle with detector's matter. In case of calorimiters, the final output of those interactions is a total energy deposited in calorimeter's fibres. In case of Zero Degree Calorimeter at ALICE, those fibres are arranged in a grid with 44×44 size. To simulate continual learning scenario, we divided input data into 5 tasks according to the input particle's energy as presented in Fig 9.6.1. Such split simulates changing conditions inside the LHC, where energy of collided beams changes between different periods of data gathering.

9.6.4. Two latents Variational Autoencoder

In the global part of our training, we rely on the regularisation of VAE's latent space. In practice, when encoding examples from distinct classes into the same latent space of VAE, we can observe that some latent variables are used to distinguish encoded class, and therefore they do not follow desired continuous distribution as observed by Tomczak and Welling (2018) and Mathieu et al. (2018). The extended experimental analysis of this phenomenon can be found in the supplementary material.

Therefore, in this work, we propose a simple disentanglement method with an additional binary latent space that addresses this problem, similar to the one introduced in Ramapuram et al. (2020). To that end, we train our encoder to encode input data characteristics into a set of continuous variables μ_c and binary variables μ_b, which are used to sample vectors λ_c and λ_b that together form λ – the input to the translator model. For the continuous variables, we follow the reparameterisation trick introduced by Kingma and Welling (2014). To sample vector λ_c, we train our encoder to generate two vectors: means μ_m and standard deviations μ_σ. Those vectors are used as parameters of Normal distribution from which we sample $\lambda_c \sim \mathcal{N}(\mu_m, \text{diag}(\mu_\sigma^2))$. For binary variables, we introduce a similar procedure based on the Gumbel softmax by Jang et al. (2016) approximation of sampling from Bernoulli distribution. Therefore, we train our encoder to produce probabilities μ_p with which we sample binary vectors $L_b \sim B(\mu_p)$. To allow generations of new data examples, for continuous values, we regularise our encoder to generate vectors λ_c from the standard normal distribution $\mathcal{N}(0, I)$ with a Kullback-Leibler divergence. For binary vectors λ_b, during inference, we approximate probabilities μ'_p with the average of probabilities μ_p for all of the examples in the train-set. We calculate μ'_p during the last epoch of the local training. Therefore, to generate new data examples we sample random continuous variables $\lambda_c \sim \mathcal{N}(0, I)$ and binary variables $\lambda_b \sim B(\mu'_p)$ and propagate them through the translator and global decoder.

9.6.5. Analysis of binary latent space

When training Variational Autoencoder with complex data distributions such as a combination of several classes, we can observe that some of the variables in the latent space do not follow desired distribution (e.g. $N(0,1)$), but instead they are used to separate latent space into different parts. In this section, we explain this behaviour on the basis of a simple example, with VAE trained on two classes from the MNIST dataset: zeros and ones. For that purpose, we analyse the latent space of the model. In Fig 9.6.5 we present distribution of continuous variables when encoding examples from separate classes. As visible, two variables (1 and 2) do not follow the standard normal distribution to which they were regularised. Instead, they are used to differentiate examples from different classes. Therefore, for certain sampled values, e.g. with variable 2 around 0, the model generates examples that are in between two classes as presented in Fig. 9.6.4. With generative replay, this problem is even more profound, since rehearsal procedure leads to error accumulation.

In this work, we propose a simple disentanglement mechanism. In the process of data encoding, we use an additional binary latent space to which the encoder can map categorical features of the input data such as distinctive classes. This simplifies encoding in standard continuous latent space in which our model does not have to separate examples from different parts of the original distribution. For comparison with standard VAE, we extended the previous model with an additional binary latent space of four binary variables. After training with the same subset of the MNIST dataset of zeros and ones, we observe that model encodes information about classes in the first binary variable as presented in Fig. 9.6.2. With such binary codes, our autoencoder does not have to separate classes in the continuous latent space, which leads to better alignment to the normal distribution as presented in Fig. 9.6.6. In Fig. 9.6.3 we show sampled generations from our disentangled representation with two latent spaces. Samples in the same column share the same continuous noise, while those in the same row have the same binary vector. Visualisation indicates that continuous features such as digit's width or rotation are shared between different binary features (column-wise), while for the same binary features (row-wise) we have only examples from the same class.

9.6.6. Visualisation of generated samples

In this section, we present additional generations from Multiband VAE. Fig. 9.6.7 shows generations from combined datasets MNIST → FashionMNIST and FashionMNIST → MNIST. Our model does not suffer from catastrophic forgetting, so pre-

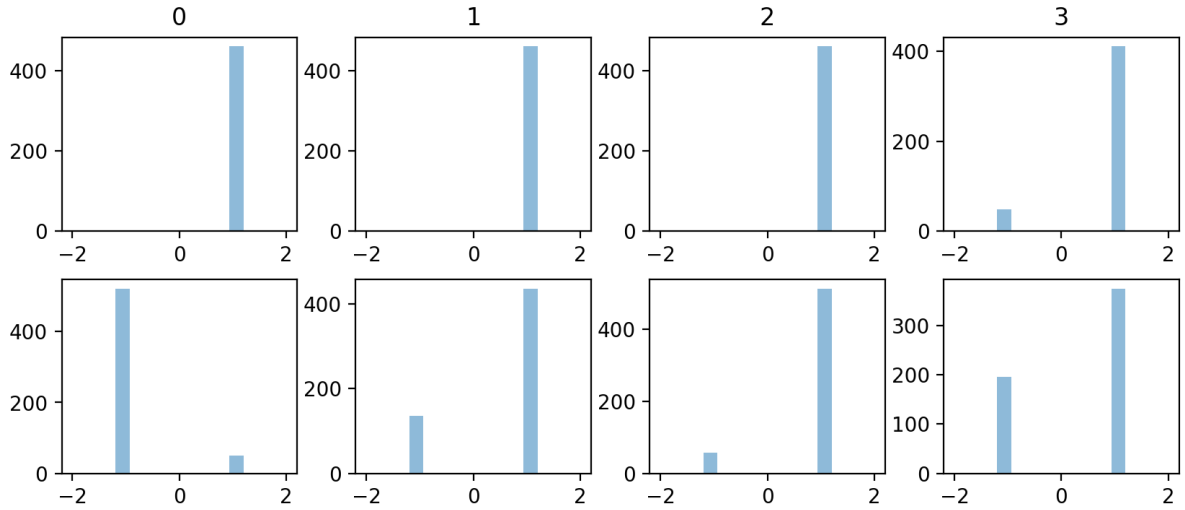


Figure 9.6.2. Binary latent space distribution of Variational Autoencoder. Sampled values for examples from encodings of class zero (top) and one (bottom). Additional binary latent space allows for simpler classes separation mostly through the first binary value for which all of the zero examples are encoded with different value than for ones.

vious generations retain their good quality even when retrained with data samples from an entirely different dataset. Moreover, it is able to identify common features between datasets, such as the thickness of generated instances or their general shape. To visualise this behaviour we generate samples from the same instance of random continuous noise (column-wise) but conditioned on different task number.

In Fig. 9.6.8a we present one more example of how our knowledge consolidation works in practice on a standard benchmark. In most cases the quality of new generations from the model retrained on top of the current global models is better than the previous one. Additionally, for some tasks, we can observe backward knowledge transfer in which training on the new task improves generations from the previous ones.

Finally in Fig. 9.6.8b we present generations on the bigger CelebA dataset. Although generations do not match those obtained from state of the art big generative models this is mainly because of the fact that we based our experiments on a shallow model similar to those used in the other approaches (VCL, hypercl, CURL) and other generative autoencoders (WAE, SAE, SWAE). Not to overshadow the main contribution, we did not use additional techniques such as deep models, Laplacian pyramid, or adversarial loss, which would improve the quality of generated samples independently from the training setup.

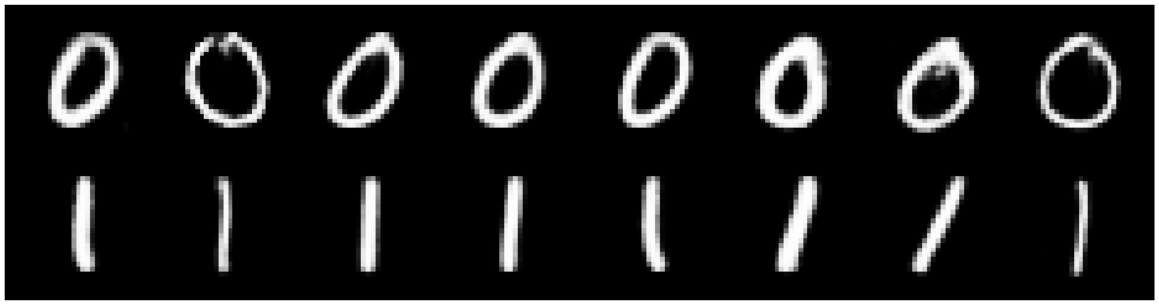


Figure 9.6.3. Examples of generations from Variational Autoencoder with binary latent space, for the same random continuous noise (per column) but opposite values for first binary variable. As visible our model well disentangles classes through binary latent space, while continuous values are still used to encode inter-class continuous features such as thickness or rotation.

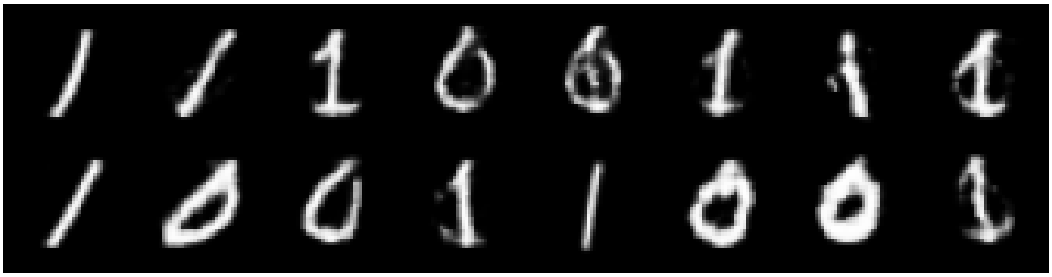


Figure 9.6.4. Examples of generations from Variational Autoencoder with no binary latent space, with variables 1 and 2 set to 0. Since model use those variables for class separation, resulting generations with sampled values around 0 are between two classes.

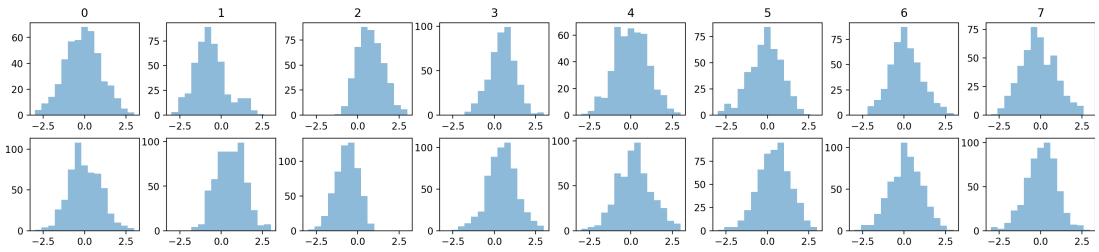


Figure 9.6.5. Latent space distribution of Variational Autoencoder trained with two separate classes. Noise embeddings for examples from class zero (top) and one (bottom). Two variables (1 and 2), do not follow standard normal distribution, but are used to differentiate examples from different classes.

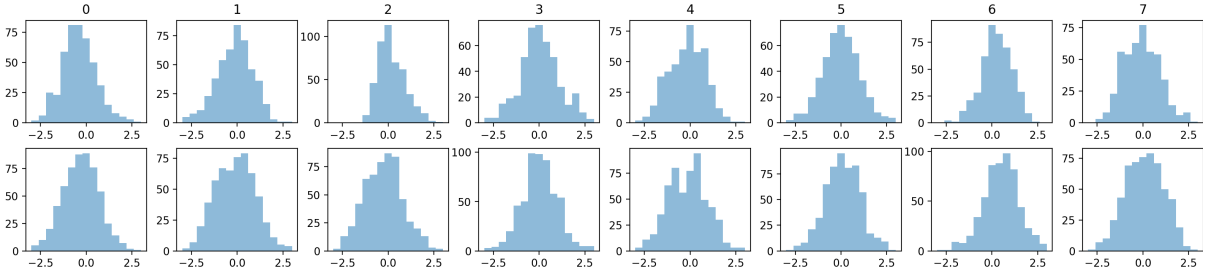
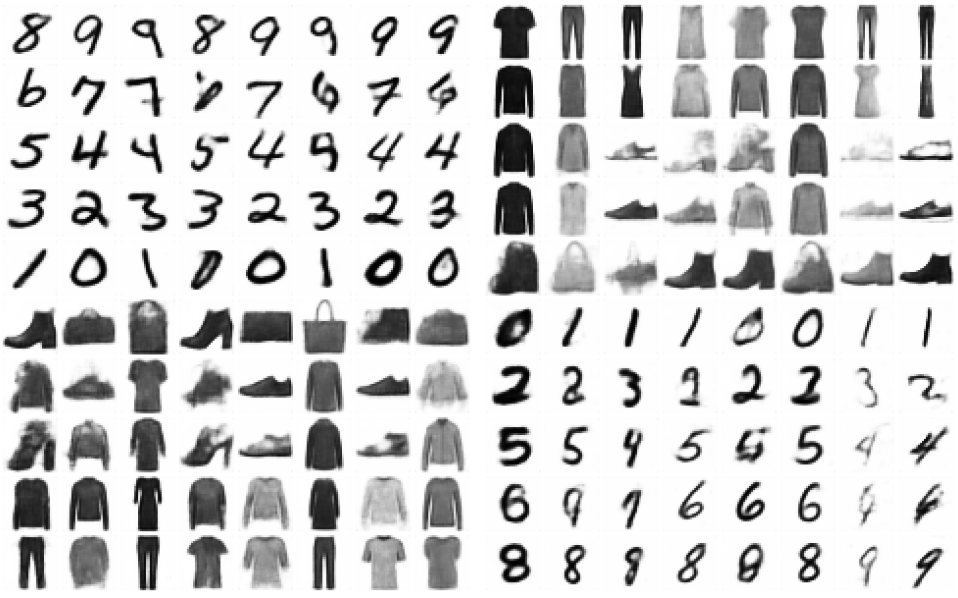
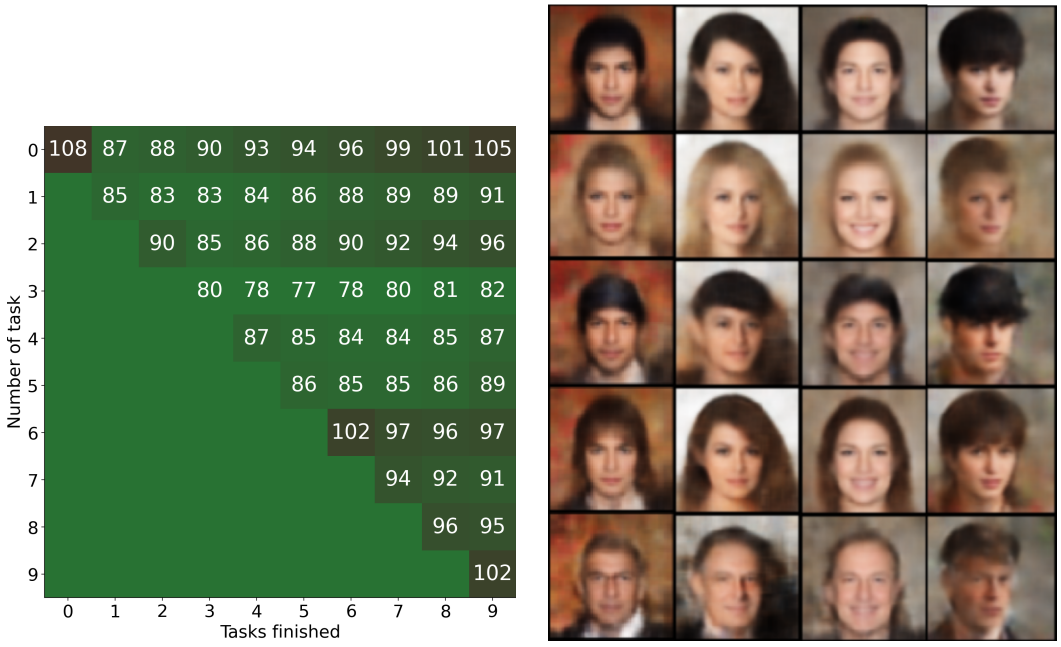


Figure 9.6.6. Latent space distribution of Variational Autoencoder with additional binary latent space trained with two separate classes. Noise embeddings for examples from class zero (top) and one (bottom). Thanks to the additional binary latent space, continual variables are better aligned to the standard normal distribution.



(a) MNIST → FashionMNIST **(b)** FashionMNIST → MNIST

Figure 9.6.7. Images generated by our Multiband VAE trained in the class incremental scenario on combined datasets MNIST → FashionMNIST (left) and FashionMNIST → MNIST (right). We generate images with the same continuous noise per column. Thanks to the proposed band arrangement procedure, we can see that even when trained on drastically different distribution our model adjusts data encodings from various tasks so that they share some common features. For example, in the first column of generations from FashionMNIST → MNIST we can observe how generations of thick black clothes correspond to the firm and bold instances of handwritten digits.



(a) **(b)**

Figure 9.6.8. FID↓ of generations from a given task of the CelebA dataset, after retraining with number of following tasks for Dirichlet $\alpha = 1$ scenario (left). Our Multiband VAE well consolidates knowledge with forward and backward knowledge transfer to generations from previous tasks when presented with new similar examples. Images generated by Multiband VAE (right) in the class incremental scenario for CelebA dataset. In the following tasks we introduce images with different hair features. In the first task we introduce photographs of people with black hair, followed by blondes, hats and brown hair. In the final task we train the model with bald and white haired people. In this visualisation we present samples with the same random continuous noise (per column) but different task index. We can observe that our Multiband VAE does not suffer from catastrophic forgetting.

10. Discussion and Final Remarks

10.1. Future Outlook of Generative Artificial Intelligence

This thesis was written at the emergence of the Large Language Models (LLM) – a new family of generative methods that have the potential to affect machine learning research significantly. Methods such as GPT-4 (OpenAI, 2023) or LLaMA (Touvron et al., 2023) have already shown that thanks to the generative modelling in a text domain, neural networks can properly encode not only the syntax of the language but also the underlying information that can be structured into knowledge. Therefore, recent systems based on those techniques, can solve complex tasks with performance comparable to humans Kojima et al. (2022). However as summarised by Mialon et al. (2023), there are still several limitations of LLMs such as lack of proper reasoning that is related to their statistical nature. Nevertheless, there are already several attempts to improve that capabilities (Creswell and Shanahan, 2022). Hence, it seems evident that generative modelling is becoming a key aspect of Machine Learning systems as a general method for learning and encoding knowledge in an unsupervised manner.

Although we have not tackled the problem of generating text in this work, we believe that the example set by LLMs will shortly proliferate to other modalities. This directly leads to the question that, in our opinion, will guide the future research in generative modelling: *Can we use generative models as a focal point of any machine learning system to represent the knowledge about the world?*

We presume that this general question will be evaluated in detail by numerous researchers in the field, as we already see a growing number of methods following this path (Dorner et al., 2022; Wu et al., 2023; Kwon et al., 2023).

10.2. Open Questions

In this thesis, we focused on one particular aspect of generative modelling: the nature of the internal latent representations created and used by those methods.

In the previous chapters, we discussed their possible structures, applications and performance in a continual learning setup. Our considerations have shed some light on the above-mentioned aspects. However, there are several straightforward questions that will guide our future research. We briefly overview them in this section.

Can we use generative model to align data representations useful for other Continual Learning tasks? In Chapter 9, we introduced Multiband VAE – a method for continual knowledge consolidation in variational autoencoder's latent space. We showed that our model can distinguish data representations from different tasks while aligning similar examples arriving to the model in a continuous stream. In this work, we focused on generative modelling and showed how we can train a VAE to continually learn the training data distribution. As future work, we will extend this idea to other ML tasks. In particular, we postulate to place the Multiband VAE in a focal point of a continually trained system – to align representations of data arriving to the model in consecutive tasks. With such structure, we will use the aligned representations in different problems such as classification, image segmentation or semi-supervised learning.

Can we benefit from data representations created by DDGMs in the continual learning setup? The exceptional performance of Deep Diffusion Generative Models can be attributed, in part, to their stable training process, which relies on large-scale datasets. Notably, the DALL-E 2 model was trained on a dataset of approximately 650 million images (Ramesh et al., 2022), and as reported by Dayma et al. (2021), the open-source implementation of this approach required 56 days of training using TPU c4 hardware. Such computational resources are often not readily available to academic institutions, smaller businesses, and independent researchers, limiting the accessibility of these models. Therefore, reducing the computational burden of diffusion models is critical to democratise their use. One promising approach to address this challenge is to leverage previously trained models and continually update them with incoming data. However, DDGMs can also suffer from catastrophic forgetting when retrained with additional data. In particular, in our preliminary work Zajac et al. (2023), we highlighted the potential challenges in continual learning of those methods. In our future research, we will continue this investigation by exploring the potential of DDGMs for learning useful data representations in a CL setup.

How can we further use the representations from DDGMs? In Chapter 6, we presented how we can benefit from data representations learned by a DDGM for

a classification task. We showed that we can improve the performance of a model over separate parameterisations by learning a joint neural network on both data distribution $p(x)$ and the marginal distribution over classes $p(y|x)$. While classification is probably one of the most common tasks in Machine Learning, numerous other applications might benefit from the robust data representations encoded in the decoder's latent space. Moreover, the initial experiments showed that the diffusion process disentangles the data representation based on their granularity. This property might be useful for image segmentation or semi-supervised learning tasks. Moreover, as a part of our method, we introduced an algorithm for counterfactual examples generations that provides partial explainability for the decisions made by a classifier. In some domains, such as medical studies, there is a great need to provide a justification or reasoning behind machine learning solutions. Joint modelling might be a helpful building block for such applications.

Can we use the diffusion process to directly learn meaningful representations? One of the essential limitations of the DDGMs is the lack of the encoder that maps the original data into low-dimensional representations. In our studies, we investigated two approaches to formulating data representations in the DDGMs, either as the intermediate steps of the diffusion process or in the latent space of the denoising model. Several recent works follow a similar idea of extracting data representations from a pre-trained model (Baranchuk et al., 2021) or by creating additional structures (Abstreiter et al., 2021). The remaining question is how to modify the diffusion process to allow for the implicit encoding-decoding process. The possible solution to this problem might be based on the work by Bansal et al. (2022), where authors introduced a diffusion process with arbitrary operations. One can imagine an operator that gradually distorts the input to the proper representation while reducing its dimensionality at the same time.

10.3. Conclusion

In this thesis, we presented an analysis of generative models from the perspective of data representations. Although this aspect of generative modelling is often marginalised as most applications focus on the property of sampling from the approximated training data distribution, we postulated that representations encoded by different generative models are robust and may be used in different downstream tasks. With a series of publications we discussed different aspects of this topic. We first analysed how different generative autoencoders and Diffusion-Based Deep Generative models encode input data into latent representations. Then, we pre-

sented a use case where representations from DDGM were used to improve the performance of a classifier through joint modelling. We also extended this analysis to semi-supervised learning or domain adaptation tasks. Finally, we showed that data representations from generative models might play an important role in continual learning, either as a practical method for the efficient storage of past data examples or an effective way for continuous knowledge consolidation.

Bibliography

Aamodt, K., Quintana, A. A., Achenbach, R., Acounis, S., Adamová, D., Adler, C., Aggarwal, M., Agnese, F., Rinella, G. A., Ahammed, Z., et al. (2008). The alice experiment at the cern lhc. *Journal of Instrumentation*, 3(08):S08002.

Abstreiter, K., Mittal, S., Bauer, S., Schölkopf, B., and Mehrjou, A. (2021). Diffusion-based representation learning. *arXiv preprint arXiv: Arxiv-2105.14257*.

Achille, A., Eccles, T., Matthey, L., Burgess, C., Watters, N., Lerchner, A., and Higgins, I. (2018). Life-long disentangled representation learning with cross-domain latent homologies. *Advances in Neural Information Processing Systems*, 31.

Ahn, H., Cha, S., Lee, D., and Moon, T. (2019). Uncertainty-based continual learning with adaptive regularization. *Advances in Neural Information Processing Systems*, 32.

Ahn, H., Kwak, J., Lim, S., Bang, H., Kim, H., and Moon, T. (2021). Ss-il: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 844–853.

Alain, G. and Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154.

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., and Page-Caccia, L. (2019). Online Continual Learning with Maximally Interfered Retrieval. In *Advances in Neural Information Processing Systems*.

Aljundi, R., Chakravarty, P., and Tuytelaars, T. (2017). Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *International Conference on Machine Learning*.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning*, pages 224–232. JMLR. org.

Arora, S. and Zhang, Y. (2017). Do gans actually learn the distribution? an empiri-

cal study. *arXiv preprint arXiv:1706.08224*.

Augustin, M., Boreiko, V., Croce, F., and Hein, M. (2022). Diffusion visual counterfactual explanations. *Advances in Neural Information Processing Systems*.

Ayinde, B. O., Inanc, T., and Zurada, J. M. (2019). Regularizing deep neural networks by enhancing diversity in feature extraction. *IEEE transactions on neural networks and learning systems*, 30(9):2650–2661.

Bansal, A., Borgnia, E., Chu, H.-M., Li, J. S., Kazemi, H., Huang, F., Goldblum, M., Geiping, J., and Goldstein, T. (2022). Cold diffusion: Inverting arbitrary image transforms without noise.

Baranchuk, D., Rubachev, I., Voynov, A., Khrulkov, V., and Babenko, A. (2021). Label-efficient semantic segmentation with diffusion models. *International Conference on Learning Representations*.

Bau, D., Zhu, J.-Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., and Torralba, A. (2019). Seeing what a gan cannot generate. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4502–4511.

Bayes, T. (1763). Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, pages 370–418.

Belouadah, E. and Popescu, A. (2019). Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 583–592.

Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013). Generalized denoising auto-encoders as generative models. *Advances in Neural Information Processing Systems*, 26.

Benny, Y. and Wolf, L. (2022). Dynamic dual-output diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11482–11491.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. (2018). Demystifying mmd gans. *International Conference on Learning Representations*.

Blaschke, T., Olivecrona, M., Engkvist, O., Bajorath, J., and Chen, H. (2018). Application of generative autoencoder in de novo molecular design. *Molecular informatics*, 37(1-2):1700123.

Bojanowski, P., Joulin, A., Lopez-Paz, D., and Szlam, A. (2017). Optimizing the latent space of generative networks. *International Conference on Machine Learning*.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *International Conference on Learning Representations*.

Brown, R. (1828). A brief account of microscopical observations made in the months of june, july and august 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. *The Philosophical Magazine*, 4(21):161–173.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *arxiv.org*.

Caccia, L., Belilovsky, E., Caccia, M., and Pineau, J. (2020). Online Learned Continual Compression with Adaptive Quantization Modules. In *Proceedings of the 37th International Conference on Machine Learning*.

Carreira-Perpinan, M. A. and Raziperchikolaei, R. (2015). Hashing with Binary Autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Chadebec, C., Vincent, L., and Allassonniere, S. (2022). Pythae: Unifying generative autoencoders in python - a benchmarking use case. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 21575–21589. Curran Associates, Inc.

Chan, C. H., Qian, K., Zhang, Y., and Hasegawa-Johnson, M. (2022). Speechsplit 2.0: Unsupervised speech disentanglement for voice conversion without tuning autoencoder bottlenecks. *Ieee International Conference On Acoustics, Speech, And Signal Processing*.

Chandra, B. and Sharma, R. K. (2014). Adaptive noise schedule for denoising autoencoder. In *International conference on neural information processing*, pages 535–542. Springer.

Chapelle, O., Scholkopf, B., and Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., and Ranzato, M. (2019). Continual Learning with Tiny Episodic Memories. In *Multi-Task and Lifelong Reinforcement Learning, Workshop at ICML*.

Chen, M., Weinberger, K., Sha, F., and Bengio, Y. (2014). Marginalized denoising auto-encoders for nonlinear representations. In *International Conference on Machine Learning*, pages 1476–1484. PMLR.

Chen, R. T., Behrmann, J., Duvenaud, D., and Jacobsen, J.-H. (2019). Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*.

Cheung, B., Terekhov, A., Chen, Y., Agrawal, P., and Olshausen, B. (2019). Superposition of many models into one. In *Advances in Neural Information Processing*

Systems.

Child, R. (2021). Very deep vaes generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*.

Chung, Y.-A., Wu, C.-C., Shen, C.-H., Lee, H.-Y., and Lee, L.-S. (2016). Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. *Interspeech 2016*, pages 765–769.

Cognigni, P., Felsenberg, J., and Waddell, S. (2018). Do the right thing: neural network mechanisms of memory formation, expression and update in drosophila. *Current opinion in neurobiology*, 49:51–58.

Colomb, J., Kaiser, L., Chabaud, M.-A., and Preat, T. (2009). Parametric and genetic analysis of drosophila appetitive long-term memory and sugar motivation. *Genes, Brain and Behavior*, 8(4):407–415.

Cresswell, J. C., Ross, B. L., Loaiza-Ganem, G., Reyes-Gonzalez, H., Letizia, M., and Caterini, A. L. (2022). Caloman: Fast generation of calorimeter showers with density estimation on learned manifolds. *arXiv preprint arXiv:2211.15380*.

Creswell, A. and Bharath, A. A. (2018). Inverting the generator of a generative adversarial network. *IEEE transactions on neural networks and learning systems*, 30(7):1967–1974.

Creswell, A. and Shanahan, M. (2022). Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300.

Dai, B. and Wipf, D. (2019). Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*.

Dayma, B., Patil, S., Cuenca, P., Saifullah, K., Abraham, T., Lê Khac, P., Melas, L., and Ghosh, R. (2021). Dalle mini.

Deja, K., Dubiński, J., Nowak, P., Wenzel, S., Spurek, P., and Trzcinski, T. (2020). End-to-end sinkhorn autoencoder with noise generator. *IEEE Access*, 9:7211–7219.

Deja, K., Kuzina, A., Trzcinski, T., and Tomczak, J. (2022a). On analyzing generative and denoising capabilities of diffusion-based deep generative models. *Advances in Neural Information Processing Systems*, 35:26218–26229.

Deja, K., Trzciński, T., Graczykowski, Ł., Collaboration, A., et al. (2018). Generative models for fast cluster simulations in the tpc for the alice experiment. In *Conference on Information Technology, Systems Research and Computational Physics*, pages 267–280. Springer.

Deja, K., Trzcinski, T., and Tomczak, J. M. (2023). Learning data representations with joint diffusion models. *arXiv preprint arXiv:2301.13622*.

Deja, K., Wawrzyński, P., Marczak, D., Masarczyk, W., and Trzciński, T. (2021). Binplay: A binary latent autoencoder for generative replay continual learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Deja, K., Wawrzyski, P., Masarczyk, W., Marczak, D., and Trzciski, T. (2022b). Multiband vae: Latent space alignment for knowledge consolidation in continual learning. In Raedt, L. D., editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2902–2908. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Deja, K. R. (2019). Using machine learning techniques for data quality monitoring in cms and alice experiments. *PoS*, page 236.

Dellacasa, G., Zhu, X., Wahn, M., Staley, F., Danielian, V., Karavicheva, T., Mikhalev, D., Carrer, N., Gheata, M., Stefanek, G., et al. (1999). Alice technical design report of the zero degree calorimeter (zdc). Technical report, ALICE.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*.

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34.

Di Sipio, R., Giannelli, M. F., Haghighat, S. K., and Palazzo, S. (2019). Dijetgan: a generative-adversarial network approach for the simulation of qcd dijet events at the lhc. *Journal of high energy physics*, 2019(8).

Ding, D., Hill, F., Santoro, A., Reynolds, M., and Botvinick, M. (2021). Attention over learned object embeddings enables complex visual reasoning. *Advances in Neural Information Processing Systems*, 34:9112–9124.

Ding, J., Condon, A., and Shah, S. P. (2018). Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature communications*, 9(1):2002.

Dorner, F. E., Peychev, M., Konstantinov, N., Goel, N., Ash, E., and Vechev, M. (2022). Human-guided fair classification for natural language processing. *arXiv preprint arXiv:2212.10154*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*.

Du, Y. and Mordatch, I. (2019). Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*.

Dudai, Y., Sher, B., Segal, D., and Yovell, Y. (1985). Defective responsiveness of adenylate cyclase to forskolin in the drosophila memory mutant rutabaga. *Jour-*

nal of neurogenetics, 2(6):365–380.

Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. (2019). Uncertainty-guided continual learning in bayesian neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 75–78.

Egorov, E., Kuzina, A., and Burnaev, E. (2021). Boovae: Boosting approach for continual learning of vae. *Advances in Neural Information Processing Systems*, 34.

Erdmann, M., Glombitza, J., and Quast, T. (2019). Precise simulation of electromagnetic calorimeter showers using a wasserstein generative adversarial network. *Computing and Software for Big Science*, 3:1–13.

Esser, P., Sutter, E., and Ommer, B. (2018). A variational u-net for conditional appearance and shape generation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Evans, L. and Bryant, P. (2008a). Lhc machine. *Journal of instrumentation*, 3(08):S08001.

Evans, L. and Bryant, P. (2008b). LHC Machine. *JINST*, 3:S08001.

Falck, F., Williams, C., Danks, D., Deligiannidis, G., Yau, C., Holmes, C. C., Doucet, A., and Willetts, M. (2022). A multi-resolution framework for u-nets with applications to hierarchical VAEs. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouve, A., and Peyré, G. (2019). Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030.

Genevay, A., Chizat, L., Bach, F., Cuturi, M., and Peyré, G. (2018). Sample complexity of sinkhorn divergences. *arXiv preprint arXiv:1810.02733*.

Genevay, A., Peyré, G., and Cuturi, M. (2017). Learning generative models with sinkhorn divergences. *arXiv preprint arXiv:1706.00292*.

Geras, K. J. and Sutton, C. (2014). Scheduled denoising autoencoders. *arXiv preprint arXiv:1406.3269*.

Giannelli, M. F., Kasieczka, G., Krause, C., Nachman, B., Salamani, D., Shih, D., and Zaborowska, A. (2022). Fast calorimeter simulation challenge.

Golkar, S., Kagan, M., and Cho, K. (2019). Continual Learning via Neural Pruning. In *Neuro AI. Workshop at NeurIPS*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014b). Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*.

Grathwohl, W., Wang, K.-C., Jacobsen, J., Duvenaud, D., Norouzi, M., and Swersky, K. (2019a). Your classifier is secretly an energy based model and you should treat it like one. *International Conference on Learning Representations*.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. (2019b). Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*.

Grathwohl, W. S., Kelly, J. J., Hashemi, M., Norouzi, M., Swersky, K., and Duvenaud, D. (2021). No {mcmc} for me: Amortized sampling for fast and stable training of energy-based models. In *International Conference on Learning Representations*.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773.

Gu, S., Chen, D., Bao, J., Wen, F., Zhang, B., Chen, D., Yuan, L., and Guo, B. (2021). Vector quantized diffusion model for text-to-image synthesis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Gupta, A., Müller, A. T., Huisman, B. J., Fuchs, J. A., Schneider, P., and Schneider, G. (2018). Generative recurrent networks for de novo drug design. *Molecular informatics*, 37(1-2):1700111.

Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. (2020). Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040.

Hayes, T. L., Krishnan, G. P., Bazhenov, M., Siegelmann, H. T., Sejnowski, T. J., and Kanan, C. (2021). Replay in deep learning: Current approaches and missing biological elements. *Neural computation*, 33(11):2908–2950.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural Collaborative Filtering. In *WWW*.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. John Wiley & Sons.

Hendrycks, D. and Gimpel, K. (2017). A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *International Conference*

on Learning Representations.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 30.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. In *Deep Learning and Representation Learning Workshop at NeurIPS*.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T. (2022). Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33.

Ho, J. and Salimans, T. (2022). Classifier-free diffusion guidance. *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

Hoffman, M. D. and Johnson, M. J. (2016). Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1.

Howard, J. N., Mandt, S., Whiteson, D., and Yang, Y. (2022). Learning to simulate high energy particle collisions from unlabeled data. *Scientific Reports*, 12(1):7567.

Hsu, T.-M. H., Qi, H., and Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.

Hsu, W.-N., Zhang, Y., and Glass, J. (2017). Learning latent representations for speech generation and transformation. *Interspeech 2017*.

Huang, C.-W., Lim, J. H., and Courville, A. C. (2021). A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34.

Huang, P. K.-M., Chen, S.-A., and Lin, H.-T. (2022). Improving conditional score-based generation with calibrated classification and joint training. In *NeurIPS 2022 Workshop on Score-Based Methods*.

Ilse, M., Tomczak, J. M., Louizos, C., and Welling, M. (2020). Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pages 322–348. PMLR.

Incerti, S., Kyriakou, I., Bernal, M., Bordage, M., Francis, Z., Guatelli, S., Ivanchenko, V., Karamitros, M., Lampe, N., Lee, S. B., et al. (2018). Geant4-dna example applications for track structure simulations in liquid water: A report from the geant4-dna project. *Medical physics*, 45(8):e722–e739.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*.

Isele, D. and Cosgun, A. (2018). Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations*.

Jebara, T. (2012). *Machine learning: discriminative and generative*, volume 755. Springer Science & Business Media.

Jin, L., Lazarow, J., and Tu, Z. (2017). Introspective classification with convolutional nets. *Advances in Neural Information Processing Systems*, 30.

Kamra, N., Gupta, U., and Liu, Y. (2017). Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*.

Kansal, R., Duarte, J., Su, H., Orzari, B., Tomei, T., Pierini, M., Touranakou, M., Gunopulos, D., et al. (2021). Particle cloud generation with message passing generative adversarial networks. *Advances in Neural Information Processing Systems*, 34.

Ke, Z., Liu, B., Ma, N., Xu, H., and Shu, L. (2021). Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34.

Kehoe, B., Patil, S., Abbeel, P., and Goldberg, K. (2015). A Survey of Research on Cloud Robotics and Automation. *IEEE T-ASE*.

Kemker, R. and Kanan, C. (2018). Fearnet: Brain-inspired model for incremental learning. *International Conference on Learning Representations*.

Khattak, G. R., Vallecorsa, S., and Carminati, F. (2018). Three dimensional energy parametrized generative adversarial networks for electromagnetic shower simulation. In *2018 25th IEEE International Conference on Image Processing (ICIP)*.

Kim, J., Kim, S., Kong, J., and Yoon, S. (2020). Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *In-*

ternational Conference On Learning Representations.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224.

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M. (2014). Semi-supervised learning with deep generative models. *Advances in Neural Information Processing Systems*, 27.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*, 29.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models. In *Advances in Neural Information Processing Systems*.

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017a). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017b). Overcoming catastrophic forgetting in neural networks. *PNAS*.

Knop, S., Spurek, P., Tabor, J., Podolak, I., Mazur, M., and Jastrzebski, S. (2020). Cramer-wold auto-encoder. *The Journal of Machine Learning Research*, 21(1):6594–6621.

Kojima, T., Gu, S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. *Neural Information Processing Systems*.

Kolouri, S., Pope, P. E., Martin, C. E., and Rohde, G. K. (2018). Sliced-wasserstein autoencoder: An embarrassingly simple generative model. *arXiv preprint arXiv:1804.01947*.

Krashes, M. J. and Waddell, S. (2008). Rapid consolidation to a radish and protein synthesis-dependent long-term memory after single-session appetitive olfactory conditioning in drosophila. *Journal of Neuroscience*, 28(12):3103–3113.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. In *Citeseer*.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. (2023). Reward design with language models. In *International Conference on Learning Representations*.

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., and Aila, T. (2019). Improved

precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32.

Kynkäänniemi, T., Karras, T., Aittala, M., Aila, T., and Lehtinen, J. (2023). The role of imagenet classes in fréchet inception distance. In *Proc. ICLR*.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning*, pages 1558–1566. PMLR.

Lasserre, J. A., Bishop, C. M., and Minka, T. P. (2006). Principled hybrids of generative and discriminative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, volume 1, pages 87–94. IEEE.

Lazarow, J., Jin, L., and Tu, Z. (2017). Introspective neural networks for generative modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2774–2783.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*.

LeCun, Y., Cortes, C., and Burges, C. (2010). MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*.

Lee, K., Xu, W., Fan, F., and Tu, Z. (2018). Wasserstein introspective neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3702–3711.

Lee, S.-H., Kim, J.-H., Chung, H., and Lee, S.-W. (2021). Voicemixer: Adversarial voice style mixup. *Advances in Neural Information Processing Systems*, 34:294–308.

Lesort, T., Caselles-Dupré, H., Garcia-Ortiz, M., Stoian, A., and Filliat, D. (2019). Generative Models from the perspective of Continual Learning. In *IJCNN*.

Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Ling, H. and Okada, K. (2006). Diffusion distance for histogram comparison. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, volume 1, pages 246–253. IEEE.

Liu, X., Wu, C., Menta, M., Herranz, L., Raducanu, B., Bagdanov, A. D., Jui, S., and de Weijer, J. v. (2020). Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 226–227.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015a). Deep learning face attributes in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3730–3738.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015b). Deep learning face attributes in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., and Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. *Nature methods*, 15(12):1053–1058.

Lopez-Paz, D. and Ranzato, M. (2017a). Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 30:6467–6476.

Lopez-Paz, D. and Ranzato, M. (2017b). Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *International Conference on Learning Representations*.

Losing, V., Hammer, B., and Wersing, H. (2017). Self-adjusting memory: How to deal with diverse drift types. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4899–4903.

Maaløe, L., Fraccaro, M., Liévin, V., and Winther, O. (2019). Biva: A very deep hierarchy of latent variables for generative modeling. *Advances in Neural Information Processing Systems*, 32.

Maaløe, L., Fraccaro, M., and Winther, O. (2017). Semi-supervised generation with cluster-aware generative models. *arXiv preprint arXiv:1704.00637*.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Mallya, A., Davis, D., and Lazebnik, S. (2018). Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. In *ECCV*.

Mallya, A. and Lazebnik, S. (2018). Packnet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Masarczyk, W., Deja, K., and Trzcinski, T. (2021). On robustness of generative representations against catastrophic forgetting. In Mantoro, T., Lee, M., Ayu, M. A., Wong, K. W., and Hidayanto, A. N., editors, *Neural Information Processing*, pages 325–333, Cham. Springer International Publishing.

Masse, N. Y., Grant, G. D., and Freedman, D. J. (2018). Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *PNAS*.

Mathieu, E., Rainforth, T., Siddharth, N., and Teh, Y. (2018). Disentangling disentanglement in variational autoencoders. *International Conference on Machine Learning*.

Mena, F. and Nanculef, R. (2019). A Binary Variational Autoencoder for Hashing. In *CIARP*.

Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., Grave, E., LeCun, Y., and Scialom, T. (2023). Augmented language models: a survey. *10.48550/arXiv.2302.07842*.

Mikuni, V. and Nachman, B. (2022). Score-based generative models for calorimeter shower simulation. *Phys. Rev. D*, 106:092009.

Müller, G. E. and Pilzecker, A. (1900). *Experimentelle beiträge zur lehre vom gedächtniss*, volume 1. JA Barth.

Mundt, M., Hong, Y., Pliushch, I., and Ramesh, V. (2023). A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*.

Mundt, M., Hong, Y. W., Pliushch, I., and Ramesh, V. (2020a). A Wholistic View of Continual Learning with Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning.

Mundt, M., Majumder, S., Pliushch, I., Hong, Y. W., and Ramesh, V. (2020b). Unified Probabilistic Deep Continual Learning through Generative Replay and Open Set Recognition.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. (2019a). Do deep generative models know what they don't know? *International Conference on Learning Representations*.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. (2019b). Hybrid models with deep and invertible features. In *International Conference on Machine Learning*, pages 4723–4732. PMLR.

Nash, C., Menick, J., Dieleman, S., and Battaglia, P. W. (2021). Generating images with sparse representations. *International Conference on Machine Learning*.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *International Conference on Learning Representations*.

Nichol, A. Q. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *In 9th ISCA Speech Synthesis Workshop*.

OpenAI (2023). Gpt-4 technical report. *ARXIV.ORG*.

Paganini, M., de Oliveira, L., and Nachman, B. (2018). Calogan: Simulating 3d high energy particle showers in multilayer electromagnetic calorimeters. *Physical*

Review D, 97(1):014021.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71.

Pascanu, R. (2021). Continual learning the challenge. ICML Workshop on Human in the Loop Learning (HILL).

Patrini, G., Berg, R. v. d., Forre, P., Carioni, M., Bhargav, S., Welling, M., Genewein, T., and Nielsen, F. (2019). Sinkhorn autoencoders. *35th Conference on Uncertainty in Artificial Intelligence*.

Perugachi-Diaz, Y., Tomczak, J., and Bhulai, S. (2021). Invertible densenets with concatenated lipswish. *Advances in Neural Information Processing Systems*, 34:17246–17257.

Pidhorskyi, S., Adjeroh, D. A., and Doretto, G. (2020). Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14104–14113.

Pol, A. A., Cerminara, G., Germain, C., Pierini, M., and Seth, A. (2019). Detector monitoring with artificial neural networks at the cms experiment at the cern large hadron collider. *Computing and Software for Big Science*, 3:1–13.

Popov, V., Vovk, I., Gogoryan, V., Sadekova, T., and Kudinov, M. (2021). Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pages 8599–8608. PMLR.

Prabhu, A., Torr, P. H., and Dokania, P. K. (2020). Gdumb: A simple approach that questions our progress in continual learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 524–540. Springer.

Qian, K., Zhang, Y., Chang, S., Hasegawa-Johnson, M., and Cox, D. (2020). Unsupervised speech decomposition via triple information bottleneck. In *International Conference on Machine Learning*, pages 7836–7846. PMLR.

Qu, H., Rahmani, H., Xu, L., Williams, B., and Liu, J. (2021). Recent advances of continual learning in computer vision: An overview. *arXiv preprint arXiv:2109.11369*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rajaraman, S., Antani, S. K., Poostchi, M., Silamut, K., Hossain, M. A., Maude, R. J., Jaeger, S., and Thoma, G. R. (2018). Pre-trained convolutional neural net-

works as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, 6:e4568.

Ramapuram, J., Gregorova, M., and Kalousis, A. (2020). Lifelong generative modeling. *Neurocomputing*, 404:381–400.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. (2019). Continual unsupervised representation learning. *Advances in Neural Information Processing Systems*, 32.

Rebuffi, S., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). iCaRL: Incremental Classifier and Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR.

Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. (2019). Experience Replay for Continual Learning. In *Advances in Neural Information Processing Systems*.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Rostami, M., Kolouri, S., and Pilly, P. K. (2019). Complementary learning for overcoming catastrophic forgetting using experience replay. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3339–3345.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive Neural Networks. arXiv:1606.04671.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*.

Sajjadi, M. S., Bachem, O., Lucic, M., Bousquet, O., and Gelly, S. (2018). Assess-

ing generative models via precision and recall. *Advances in Neural Information Processing Systems*.

Sakurada, M. and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pages 4–11.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. *Advances in Neural Information Processing Systems*, 29.

Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*.

Scardapane, S., Uncini, A., et al. (2020). Pseudo-rehearsal for continual learning with normalizing flows. In *4th Lifelong Machine Learning Workshop at ICML 2020*.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems*.

Shmelkov, K., Schmid, C., and Alahari, K. (2018). How good is my gan? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229.

Sietsma, J. and Dow, R. J. (1991). Creating artificial neural networks that generalize. *Neural networks*, 4(1):67–79.

Simidjievski, N., Bodnar, C., Tariq, I., Scherer, P., Andres Terre, H., Shams, Z., Jamnik, M., and Liò, P. (2019). Variational autoencoders for cancer data integration: design principles and computational practice. *Frontiers in genetics*, 10:1205.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

Song, J., Meng, C., and Ermon, S. (2020a). Denoising diffusion implicit models. In *International Conference on Learning Representations*.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Stoianov, I., Maisto, D., and Pezzulo, G. (2022). The hippocampal formation as a hierarchical generative model supporting generative replay and continual learning. *Progress in Neurobiology*, 217:102329.

Strub, F., Gaudel, R., and Mary, J. (2016). Hybrid recommender system based on autoencoders. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 11–16.

Strulab, D., Santin, G., Lazaro, D., Breton, V., and Morel, C. (2003). Gate (geant4 application for tomographic emission): a pet/spect general-purpose simulation platform. *Nuclear Physics B-Proceedings Supplements*, 125:75–79.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the inception architecture for computer vision. *Computer Vision and Pattern Recognition*.

Szymczak, P., Możejko, M., Grzegorzek, T., Jurczak, R., Bauer, M., Neubauer, D., Sikora, K., Michalski, M., Sroka, J., Setny, P., et al. (2022). Discovering highly potent antimicrobial peptides with deep generative model hydramp. *bioRxiv*, pages 2022–01.

Tashiro, Y., Song, J., Song, Y., and Ermon, S. (2021). Csdi: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, volume 34, pages 24804–24816. Curran Associates, Inc.

Tempel, B. L., Bonini, N., Dawson, D. R., and Quinn, W. G. (1983). Reward learning in normal and mutant drosophila. *PNAS*, 80(5):1482–1486.

Thai, A., Stojanov, S., Rehg, I., and Rehg, J. M. (2021). Does continual learning= catastrophic forgetting. *arXiv preprint arXiv:2101.07295*.

Thandiackal, K., Portenier, T., Giovannini, A., Gabrani, M., and Goksel, O. (2021). Generative feature-driven image replay for continual learning. *arXiv preprint arXiv:2106.05350*.

Theis, L., Shi, W., Cunningham, A., and Huszár, F. (2017). Lossy image compression with compressive autoencoders. *International Conference on Learning Representations*.

Tilaro, F., Bradu, B., Gonzalez-Berges, M., Roshchin, M., and Varela, F. (2018). Model Learning Algorithms for Anomaly Detection in CERN Control Systems. In *ICALEPCS*.

Tinchev, G., Czarnowska, M., Deja, K., Yanagisawa, K., and Cotescu, M. (2023). Modelling low-resource accents without accent-specific tts frontend. *arXiv preprint arXiv:2301.04606*.

Tits, N., Wang, F., El Haddad, K., Pagel, V., and Dutoit, T. (2019). Visualization and interpretation of latent spaces for controlling expressive speech synthesis through audio analysis. *Proc. Interspeech 2019*, pages 4475–4479.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2017). Wasserstein auto-encoders. *International Conference on Learning Representations*.

Tomczak, J. and Welling, M. (2018). Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223.

Tomczak, J. M. (2022). *Deep Generative Modeling*. Springer Cham.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models. *ARXIV.ORG*.

Tschannen, M., Bachem, O., and Lucic, M. (2018). Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*.

Tulyakov, S., Fitzgibbon, A., and Nowozin, S. (2017). Hybrid vae: Improving deep generative models using partial observations. *arXiv preprint arXiv:1711.11566*.

Tzen, B. and Raginsky, M. (2019). Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*.

Vahdat, A. and Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679.

Vahdat, A., Kreis, K., and Kautz, J. (2021). Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34.

Van de Ven, G. M., Siegelmann, H. T., and Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):4069.

van de Ven, G. M. and Tolias, A. S. (2018). Generative replay with feedback connections as a general strategy for continual learning. arXiv:1809.10635.

Van de Ven, G. M. and Tolias, A. S. (2019). Three scenarios for continual learning. *NeurIPS - Continual Learning workshop*.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017). Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems*.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408.

von Oswald, J., Henning, C., Sacramento, J., and Grewe, B. F. (2019). Continual learning with hypernetworks. In *International Conference on Learning Representations*.

Wang, D., Deng, L., Yeung, Y. T., Chen, X., Liu, X., and Meng, H. (2021a). Vqmivc: Vector quantization and mutual information-based unsupervised speech repre-

sentation disentanglement for one-shot voice conversion. *Interspeech*.

Wang, L., Lei, B., Li, Q., Su, H., Zhu, J., and Zhong, Y. (2021b). Triple-memory networks: A brain-inspired method for continual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):1925–1934.

Wang, S., Li, X., Sun, J., and Xu, Z. (2021c). Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 184–193.

Wang, Y., Yao, H., and Zhao, S. (2016). Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242.

Wang, Z., Simoncelli, E. P., and Bovik, A. C. (2003). Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. IEEE.

Wehenkel, A. and Louppe, G. (2021). Diffusion priors in variational autoencoders. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

Wei, R. and Mahmood, A. (2020). Recent advances in variational autoencoders with representation learning for biomedical informatics: A survey. *Ieee Access*, 9:4939–4956.

Wen, Y., Tran, D., and Ba, J. (2020). BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning. In *International Conference on Learning Representations*.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. (2020). Supermasks in Superposition. In *Advances in Neural Information Processing Systems*.

Wu, C., Herranz, L., Liu, X., Wang, Y., van de Weijer, J., and Raducanu, B. (2018). Memory replay gans: Learning to generate new categories without forgetting. In *Advances in Neural Information Processing Systems*.

Wu, S., Wang, J., Ping, W., Nie, W., and Xiao, C. (2023). Defending against adversarial audio via diffusion model. In *International Conference on Learning Representations*.

Xiang, Y., Fu, Y., Ji, P., and Huang, H. (2019). Incremental Learning Using Conditional Adversarial Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747.

Xiao, Z., Yan, Q., and Amit, Y. (2019). Generative latent flow. *arXiv preprint arXiv:1905.10485*.

Xu, J. and Zhu, Z. (2018). Reinforced Continual Learning. In *Advances in Neural*

Information Processing Systems.

Yan, S., Xie, J., and He, X. (2021). Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023.

Yang, B., Luo, W., and Urtasun, R. (2018). Pixor: Real-time 3d Object Detection from Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yang, L.-C., Chou, S.-Y., and Yang, Y.-H. (2017). Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*.

Yang, W., Kirichenko, P., Goldblum, M., and Wilson, A. G. (2022a). Chroma-vae: Mitigating shortcut learning with generative classifiers. *Advances in Neural Information Processing Systems*.

Yang, X. and Ji, S. (2021). Jem++: Improved techniques for training jem. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6494–6503.

Yang, X., Shih, S.-M., Fu, Y., Zhao, X., and Ji, S. (2022b). Your vit is secretly a hybrid discriminative-generative diffusion model. *arXiv preprint arXiv:2208.07791*.

Ye, F. and Bors, A. G. (2020). Learning latent representations across multiple data domains using lifelong vaegan. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 777–795. Springer.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). Lifelong Learning with Dynamically Expandable Networks. In *International Conference on Learning Representations*.

Yuan, S., Cheng, P., Zhang, R., Hao, W., Gan, Z., and Carin, L. (2021). Improving zero-shot voice style transfer via disentangled representation learning. In *International Conference on Learning Representations*.

Zajac, M., Deja, K., Kuzina, A., Tomczak, J. M., Trzciski, T., Shkurti, F., and Mio, P. (2023). Exploring continual learning of diffusion models. *arXiv preprint arXiv: Arxiv-2303.15342*.

Zamorski, M., Zieba, M., Klukowski, P., Nowak, R., Kurach, K., Stokowiec, W., and Trzcinski, T. (2020). Adversarial Autoencoders for Compact Representations of 3D Point Clouds. *Comput. Vis. Image Underst.*

Zeng, G., Chen, Y., Cui, B., and Yu, S. (2019). Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372.

Zenke, F., Poole, B., and Ganguli, S. (2017a). Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995.

PMLR.

Zenke, F., Poole, B., and Ganguli, S. (2017b). Continual Learning Through Synaptic Intelligence. In *Proceedings of the 34th International Conference on Machine Learning*.

Zhang, C., Song, N., Lin, G., Zheng, Y., Pan, P., and Xu, Y. (2021). Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12455–12464.

Zhang, Q. and Zhang, L. (2018). Convolutional adaptive denoising autoencoders for hierarchical feature extraction. *Frontiers of Computer Science*, 12(6):1140–1148.

Zhou, C. and Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674.

Zhu, J., Zhao, D., Zhang, B., and Zhou, B. (2022). Disentangled inference for gans with latently invertible autoencoder. *International Journal of Computer Vision*.

Zhu, J.-Y., Krähenbühl, P., Shechtman, E., and Efros, A. A. (2016). Generative visual manipulation on the natural image manifold. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 597–613. Springer.

Ziegler, Z. M. and Rush, A. M. (2019). Latent normalizing flows for discrete sequences. *International Conference on Machine Learning*.

Zoglauer, A., Andritschke, R., and Schopper, F. (2006). Megalib–the medium energy gamma-ray astronomy library. *New Astronomy Reviews*, 50(7-8):629–632.