KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Unsupervised Representation Learning and Health Insurance Anomaly Detection

Hannes De Meulemeester

Supervisor:
Prof. dr. ir. Bart De Moor

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

February 2024

Unsupervised Representation Learning and Health Insurance Anomaly Detection

Hannes DE MEULEMEESTER

Examination committee:
Em. prof. dr. ir. Jean-Pierre Celis, chair
Prof. dr. ir. Bart De Moor, supervisor
Prof. dr. ir. dr. Frank De Smet
Prof. dr. ir. Johan Suykens
Prof. dr. ir. Hendrik Blockeel
Prof. dr. ir. Tinne Tuytelaars
Prof. dr. Tijl De Bie
(Ghent University)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

February 2024

© 2024 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Hannes De Meulemeester, Kasteelpark Arenberg 10, bus 2446, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Preface

This thesis is the result of five years of research under the supervision of Prof. Bart De Moor in the STADIUS group at the Department of Electrical Engineering at KU Leuven. This work would not have been possible without the help, friendship and support of a great many people.

First, I would like to thank my supervisor, Prof. De Moor, for the incredible opportunity. His guidance and expertise have allowed me to successfully navigate the academic world. I would especially like to thank him for the freedom he offered in pursuing my own research avenues, which allowed me to define myself as a researcher.

Next, I would like to thank the members of the examination committee, Prof. Jean-Pierre Celis, Prof. Frank De Smet, Prof. Johan Suykens, Prof. Hendrik Blockeel, Prof. Tinne Tuytelaars and Prof. Tijl De Bie, for reading this thesis and providing helpful feedback which helped polish the text.

Further, I would like to offer my sincere thanks to Prof. Suykens. When I first talked to you about being interested in your research and wanting to work more closely together, you welcomed me with open arms and treated me as a member of your own team. Your vast knowledge of machine learning has been an invaluable resource to me and I am grateful to have received the opportunity to learn and work with you.

In addition, I would like to thank Prof. Frank De Smet for the wonderful support and cooperation during the work with the CM. You helped me navigate the complicated world of healthcare insurance and made me feel like part of the team at the CM.

I would like to thank all of my colleagues at STADIUS for the many interesting and fun times at work, during coffee breaks, lunch or when going out for drinks. You created a motivating and wonderful environment for everyone.

Michaël, you were a role model for me and not a single discussion went by

without learning something from you. You helped build up everyone around you as a researcher, just as you did for me, for which I will always admire you.

Joachim, together with Michaël, you were the first person I collaborated with during my PhD. We were able to bounce ideas of each other amazingly well and our many talks and discussions were an inspiration during our years working together.

Arun, thank you for all the amazing cooperations and talks both inside and outside of work. One of the most difficult parts of my PhD was trying to figure out the keybindings on your laptop.

Henri, thank you for all the good times and interesting work we did together. I will always fondly remember the conversations we had in the Ardennes until the early hours.

Finally, I would like to thank my family. Mama and papa for their unwavering support throughout my personal and academic life. None of this would have been possible without you. Nonkel Koen, who's own PhD defense left an impression on me when I was young and was part of the motivation in starting this journey.

A final thank you to friends, family, colleagues and everyone else who offered their support, aid or simply a word of wisdom that helped me along the way.

Abstract

Machine learning has broken into every aspect of our lives and is becoming indispensable in our current society. We come into contact with machine learning models when using smartphones, shopping online, driving cars and many other ways without even knowing it. The wide use of these models is driven by the breakthroughs that machine learning is making in almost every imaginable field from industrial processes to healthcare to science and even art.

One of the most important reasons behind the advances in machine learning is due to their increasingly powerful representation learning capabilities. The most well-known type of representation learning is a neural network, which is a powerful type of model that can automatically extract features out of input data. However, one has little control over the types of features that a neural network learns. This motivates research into learning representations that can capture prior knowledge, offer insight into the model and can improve the learning process. In this work, a number of unsupervised representation learning methods are discussed to increase performance, tackle biases, improve diversity and explainability of machine learning models. Finally, a novel representation learning technique in combination with other state-of-the-art methods are employed to create a system that can help health insurance experts in detecting anomalous profiles.

Machine learning models generally minimize an empirical expectation of the data. The same is true for generative models such as generative adversarial networks. These models can overlook minority modes or learn unwanted biases when attempting to learn the data distribution. We propose two methods to promote diversity in generative adversarial networks. The first is to augment the standard loss by the Bures distance between the covariance matrices of the generated and real data to more closely match their distributions. The second method uses a diversity sampling technique using leverage score sampling such that a higher sampling probability is given to more unique samples during training.

Restricted kernel machines are a novel type of machine learning method where the training process involves solving a kernel PCA problem. This allows a model to learn interpretable features from the data. We extend this model to time-series data in two different ways such that it can learn time-series features and can predict unseen future values.

Finally, the problem of unsupervised health insurance outlier detection is tackled. In this setting, the data is of a tabular format that contains both real and categorical features. Generally, categorical features cannot be used with most machine learning methods. Therefore, a method to create categorical embeddings is discussed. It is shown that these embeddings are able to capture useful relations between the variables and increase the performance of machine learning methods. Using this method, and other state-of-the-art techniques in machine learning, a workflow is developed for unsupervised anomaly detection for health insurance in collaboration with the largest health insurance fund in Belgium. The workflow is able to help experts in finding anomalous practitioner profiles and aids them in investigating such practitioners by offering explanations of the machine learning predictions.

Beknopte samenvatting

Machine learning is doorgedrongen in elk aspect van ons leven en is onmisbaar aan het worden in de huidige samenleving. We komen in aanraking met machine learning-modellen bij het gebruik van onder andere smartphones, online winkelen, autorijden en vele andere manieren zonder er bij stil te staan. Het brede gebruik van deze modellen kan worden verklaard door de doorbraken die machine learning maakt in bijna elk denkbaar gebied, van industriële processen tot gezondheidszorg tot wetenschap en zelfs kunst.

Een van de belangrijkste redenen van de vooruitgang in machine learning is te danken aan het steeds krachtiger wordende representatieleren. Het bekendste type van representatieleren is een neuraal netwerk, krachtige modellen die automatisch eigenschappen uit invoergegevens kunnen extraheren. Men heeft echter weinig controle over de soorten representaties die een neuraal netwerk leert. Dit motiveert onderzoek naar representatieleren die voorkennis kan integreren, inzicht in het model kan bieden en het leerproces kan verbeteren. In dit werk worden een aantal unsupervised machine learning-methoden voor representatieleren besproken om de prestaties te verbeteren, bias aan te pakken, de diversiteit en de verklaarbaarheid van machine learning-modellen te verbeteren. Ten slotte wordt een nieuwe techniek in representatieleren in combinatie met andere state-of-the-art methoden gebruikt om een systeem te creëren dat zorgverzekeraars helpt bij het opsporen van afwijkende profielen.

Machine learning-modellen minimaliseren over het algemeen een empirische verwachting van de data. Hetzelfde geldt voor generatieve modellen zoals generative adversarial networks. Deze modellen kunnen minderheidsmodi over het hoofd zien of ongewenste vooroordelen opnemen wanneer ze proberen de gegevensdistributie te leren. We stellen twee methoden voor om diversiteit in generative adversarial networks te bevorderen. De eerste is om de standaard verliesfunctie te vermeerderen met de Bures-afstand tussen de covariantiematrices van de gegenereerde en echte gegevens om hun distributies beter te laten overeenkomen. De tweede methode maakt gebruik van een

diversiteitssteekproef zodat een hogere bemonsteringskans wordt gegeven aan meer unieke datapunten tijdens de training.

Restricted kernel machines zijn een nieuw type machine learning-methode waarbij het trainingsproces, het oplossen van een kernel-PCA-probleem inhoudt. Hierdoor kan een model interpreteerbare kenmerken uit de gegevens leren. We breiden dit model op twee verschillende manieren uit naar tijdreeksdata, zodat het kenmerken, specifiek aan tijdsreeksen, kan leren en ongeziene toekomstige waarden kan voorspellen.

Ten slotte wordt het probleem van de detectie van anomalieën in de zorgverzekering aangepakt aan de hand van unsupervised machine learning. In deze opstelling zijn de gegevens in de vorm van tabulaire data die zowel reële als categorische variabelen bevat. Over het algemeen kan categorische data niet worden gebruikt bij de meeste machine learning-methoden. Daarom wordt een methode besproken om categorische inbeddingen te maken. Het is aangetoond dat deze inbeddingen in staat zijn om nuttige relaties tussen de variabelen vast te leggen en de prestaties van machine learning-methoden te verbeteren. Met behulp van deze methode en andere state-of-the-art technieken in machine learning, wordt in samenwerking met het grootste ziekenfonds van België een workflow ontwikkeld voor anomaliedetectie voor de zorgverzekeringsgegevens. De workflow kan experts helpen bij het vinden van afwijkende beroepsprofielen en ondersteunt hen bij het onderzoeken van betreffende beroepsbeoefenaars door toelichting te geven over de machine learning-voorspellingen.

List of Abbreviations

ABOD Angle-base Outlier Detection.

AI Artificial Intelligence.

ARMA Autoregressive Moving Average.

AUROC Area Under the ROC curve.

CBOW Continuous Bag of Words.

CM Christian Health Insurance Fund.

COPOD Copula-Based Outlier Detection.

CVaR Conditional Value-at-Risk.

DCGAN Deep Convolutional Generative Adversarial Network.

DeepSVDD Deep Support Vector Data Description.

DPP Determinantal Point Process.

ECOD Empirical-Cumulative-distribution-based Outlier Detection.

FID Fréchet Inception Distance.

FPS Federal Public Service.

GAN Generative Adversarial Network.

GDP Gross Domestic Product.

GloVe Global Vectors for Word Representation.

GP General Practitioner.

GPU Graphics Processing Unit.

IF Isolation Forest.

INAMI Institut national d'assurance maladie-invalidité.

INNE Isolation-based Nearest-Neighbor Ensembles.

IS Inception Score.

KL Kullback–Leibler.

KNN K Nearest Neighbors.

KPCA Kernel Principal Component Analysis.

KRLS Kernel Recursive Least-Squares.

LODA Lightweight On-line Detector of Anomalies.

LOF Local Outlier Factor.

LS-SVM Least-Squares Support Vector Machines.

LSA Latent Semantic Analysis.

LSTM Long-Short Term Memory network.

LUNAR Learnable Unified Neighborhood-based Anomaly Ranking.

MMD Maximum Mean Discrepancy.

MV-RKM Multi-View Restricted Kernel Machine.

NAR Non-linear Auto-Regressive.

NLP Natural Language Processing.

OCSVM One-Class Support Vector Machine.

PCA Principal Component Analysis.

PDF Probability Density Function.

RBM Restricted Boltzmann Machine.

RIZIV Rijksinstituut voor ziekte- en invaliditeitsverzekering.

RKHS Reproducing Kernel Hilbert Space.

RKM Restricted Kernel Machine.

RLS Ridge Leverage Score.

RNN Recurrent Neural Network.

ROC Receiver Operating Characteristic.

RRKM Recurrent Restricted Kernel Machine.

SHAP Shapley additive explanations.

SOD Subspace Outlier Detection.

SVD Singular Value Decomposition.

SVM Support Vector Machine.

SWD Sliced Wasserstein Distance.

t-SNE t-Distributed Stochastic Neighbor Embedding.

TRBM Temporal Restricted Boltzmann Machine.

URF Unsupervised Random Forest.

VAE Variational Autoencode.

WGAN Wasserstein Generative Adversarial Network.

List of Symbols

a	Scalar a
$\boldsymbol{a}$	Vector $\boldsymbol{a}$
$\boldsymbol{A}$	Matrix $\boldsymbol{A}$
Tr	Matrix trace
$\|\cdot\|_F$	Frobenius norm
$.^\top$	Transpose
diag	Diagonal of a matrix
$k(\cdot,\cdot)$	Kernel function
$\boldsymbol{K}$	Kernel matrix
$\mathbb{R}$	The set of real numbers
$E[.]$	Expectation
$\|\cdot\|_2$	L_2 norm
sup	Supremum
$\mathbb{I}_n$	Identity matrix of size $n \times n$
$\phi(\cdot)$	Feature map
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$

Contents

Abstract iii

Beknopte samenvatting v

List of Abbreviations ix

List of Symbols xi

Contents xiii

List of Figures xvii

List of Tables xxiii

1 Introduction 1
- 1.1 Unsupervised Representation Learning . . . 2
- 1.2 Background . . . 3
  - 1.2.1 Generative Adversarial Networks . . . 3
  - 1.2.2 Restricted Kernel Machines . . . 7
  - 1.2.3 Health Insurance Overconsumption and Fraud . . . 12
- 1.3 Research Aims . . . 14
- 1.4 Thesis Outline . . . 16

2 Diversity in Generative Adversarial Networks 19
- 2.1 The Bures Metric for Generative Adversarial Networks . . . 21
  - 2.1.1 Introduction . . . 21
  - 2.1.2 Method . . . 23
  - 2.1.3 Empirical Evaluation of Mode Collapse . . . 27
  - 2.1.4 Real Images . . . 31
  - 2.1.5 High Quality Generation using a ResNet Architecture . 33
  - 2.1.6 Conclusion . . . 34

2.2 Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks . . . 34
2.2.1 Introduction . . . 35
2.2.2 Sampling with Ridge Leverage Scores . . . 38
2.2.3 Numerical experiments . . . 42
2.2.4 Conclusion . . . 48

3 Restricted Kernel Machines for Time Series 51
3.1 Recurrent Restricted Kernel Machines for Time-series Forecasting 52
3.1.1 Introduction . . . 52
3.1.2 Recurrent Restricted Kernel Machines . . . 52
3.1.3 Prediction . . . 54
3.1.4 Experiments . . . 56
3.1.5 Conclusion . . . 58
3.2 Multi-view Kernel PCA for Time series Forecasting . . . 59
3.2.1 Introduction . . . 59
3.2.2 Related Works . . . 60
3.2.3 Proposed Model . . . 62
3.2.4 Experiments . . . 68
3.2.5 Conclusion . . . 72

4 Unsupervised Embeddings for Categorical Variables 73
4.1 Introduction . . . 74
4.2 Natural Language Processing . . . 75
4.2.1 CBOW and Skip-gram . . . 76
4.2.2 LSA . . . 76
4.2.3 GloVe . . . 77
4.3 Methodology . . . 77
4.4 Experiments . . . 79
4.4.1 Simple Example . . . 79
4.4.2 Nursery Data . . . 80
4.4.3 Classification Performance . . . 83
4.5 Conclusion . . . 86
4.6 Future Work . . . 87

5 Explainable Unsupervised Anomaly Detection for Healthcare Insurance Data 89
5.1 Background . . . 90
5.1.1 Related Work . . . 92
5.2 Methods . . . 93
5.2.1 Preprocessing . . . 94
5.2.2 Models . . . 97
5.2.3 Explainability . . . 98

5.3 Results and Discussion . . . 102
5.3.1 Belgian Health Insurance Data . . . 102
5.3.2 Experiment Results . . . 102
5.3.3 Explainability . . . 104
5.4 Conclusion . . . 105

6 Conclusion **107**

A The Bures metric for generative adversarial networks **111**
A.1 Details of the theoretical results . . . 111
A.2 Additional Experiments . . . 112
A.2.1 Alternating training . . . 112
A.2.2 Another matrix norm . . . 112
A.2.3 Timings . . . 113
A.2.4 Best inception scores achieved with DCGAN architecture 113
A.2.5 Influence of the number of convolutional layers for DCGAN architecture . . . 114
A.3 Additional Figures . . . 118
A.3.1 Training visualization at different steps for the artificial data sets . . . 118
A.3.2 Generated images of the trained GANs . . . 118
A.4 Architectures . . . 125
A.4.1 Synthetic Architectures . . . 125
A.4.2 Stacked MNIST Architectures . . . 125
A.4.3 CIFAR-10 and 100 DCGAN Architectures . . . 129
A.4.4 STL-10 DCGAN Architectures . . . 130
A.4.5 ResNet Architectures . . . 131

B Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks **134**
B.1 Organization . . . 134
B.2 Extra table UNBALANCED MNIST dataset . . . 134
B.3 Effect of the regularization parameter in RLS . . . 135
B.3.1 Synthetic data . . . 135
B.3.2 Unbalanced MNIST . . . 136
B.4 Timings . . . 139
B.5 Distribution of generated samples for the artificial datasets . . . 139
B.6 Architectures, algorithms, and settings . . . 140

C Multi-view Kernel PCA for Time series Forecasting **144**
C.1 Note on kernel centering . . . 144
C.2 Datasets and Hyperparameters . . . 145
C.3 Least-Squares Support Vector Machine . . . 145

D Explainable Unsupervised Anomaly Detection for Healthcare Insurance Data **147**
D.1 Shapley Values 147
D.2 Shapley Values for Machine Learning 148
D.3 General Practitioner Data used during the Experiments 150

Bibliography **153**

Curriculum **177**

List of publications **179**

List of Figures

1.1 The generator G maps samples from the known latent distribution $\mathcal{Z}$ to the input space $\mathcal{X}$. The discriminator D takes as input samples from the input space $\mathcal{X}$ and classifies these samples into fake, generated by G, or real, belonging to the original dataset. 5
1.2 An extreme example of mode collapse in GANs. On the left a sample from the original MNIST dataset can be observed while on the right samples generated by the GAN. The generator has learned only a single mode of the dataset corresponding to the number 1. 7
1.3 Figure taken from the WHO Global Health Expenditure Database [231]. Globally, from low to high income countries, the healthcare spending as a part of their respective GDP has been steadily rising since the year 2000. 12

2.1 Figure accompanying Table 2.1, the progress of BuresGAN on the synthetic examples. Each column shows 3k samples from the training of the generator in blue and 3k samples from the true distribution in green. 28
2.2 Generated samples from a trained BuresGAN, with a DCGAN architecture. 29
2.3 Images generated by BuresGAN with a ResNet architecture for CIFAR-10 (left) and STL-10 (right) data sets. The STL-10 samples are full-sized 96×96 images. 32

2.4 Probability Density Function (orange) and RLS of a sample of this PDF (blue). We take the motivating example from [243], which consists of a 1D target PDF p with 1 majority mode and 2 minority modes: $p = 0.9 \cdot \mathcal{N}(0,1) + 0.05 \cdot \mathcal{N}(10,1) + 0.05 \cdot \mathcal{N}(-10,1)$ given in orange. The RLS distribution is calculated using a Gaussian kernel with $\sigma = 3$ and $\gamma = 10^{-3}$. When sampling a mini-batch from the PDF p, the side modes can be missed. This is resolved by sampling from the RLS distribution. 36
2.5 RLS distribution using a Gaussian feature map with $\sigma = 0.15$ and regularization $\gamma = 10^{-3}$ on the unbalanced RING (left) and GRID (right) data. The darker the shade, the higher the RLS. Dark modes correspond to minority modes. 39
2.6 Visualization of the generation quality on RING and GRID. Each column shows 2.5k samples from the trained generator in blue and 2.5k samples from the true distribution in green. The vanilla GAN (first and third) does not cover the minority modes. This is not the case for the RLS BuresGAN Discr. (second and fourth). 43
2.7 RING. Number of training samples in each mode for the RING dataset (left) Generated samples in each mode by a vanilla GAN (middle). Generated samples by RLS BuresGAN Discr. (right). A rebalancing effect is visible. 45
2.8 Generated images from UNBALANCED 012-MNIST by a vanilla GAN (first), by RLS BuresGAN Discr. (second) and generated images from UNBALANCED MNIST by a vanilla GAN (third), by RLS BuresGAN Class. (fourth). The minority digits are generated more frequently in the proposed methods that include RLS sampling. 46
2.9 Generated images from 06-CIFAR10 by a StyleGAN2 + Aug. (left) and by RLS StyleGAN2 + Aug. (right). Including RLS sampling promotes sampling from the minority class. Generated samples classified as planes are marked by a red border. 48

3.1 Dependency graph of the Recurrent RKM model's *training* (3.3) and *prediction* (3.8) scheme for $a_{t,l} = 1$ if $l = 1$ and $a_{t,l} = 0$ otherwise, and a linear kernel on $\mathscr{X}$. 54
3.2 Training and predicted latent variables of a sinusoidal data set. 56
3.3 Ablation study on the Santa Fe laser data set where the blue curve is the ground-truth and the green curve is the prediction. 57
3.4 MV-RKM on sin wave. The embedded data points are *temporally* de-correlated in the embedding space. 69

3.5 Predictions (left) and components (right) for the sum of sine and SantaFe experiments. More components correspond to a better prediction. A lower lag for the same parameters generally generate higher frequency components and more components are required for the same performance. 70
3.6 Prediction error with different number of components on the sum of sines (left) and SantaFe (right) datasets. The blue curve indicates the prediction of the model for the best lag. The gray curve indicates the mean prediction error over multiple lag values with corresponding standard deviation. It can be seen that more components indicate a better reconstruction until a plateau is reached. 71

4.1 The t-SNE visualisations of respectively CBOW, Skip-gram, LSA and GloVe for the simple example. CBOW and Skip-gram perform well at representing the categorical variables, shown by the clear variable clusters. LSA is able to extract all dependencies, as indicated by the closeness of the related values in embedding space, but has less distinct clusters. GloVe performs well at both tasks. 81
4.2 The PCA visualisations of the two best performing models, LSA and GloVe, for the simple example. The visualisations show the same relations as those that can be observed on the t-SNE plots: Both LSA and GloVe are able to extract the relations but GloVe is better at constructing clusters of the categorical values. . . . 82
4.3 The t-SNE visualisations of respectively CBOW, Skip-gram, LSA and GloVe for the nursery data set. CBOW and Skip-gram do well at representing the categorical variables, indicated by their corresponding distinct clusters. LSA is able to extract more relations than CBOW and Skip-gram: some attributes of the same hierarchical level are closer together in the embeddings space. GloVe performs well at both. 83

5.1 The general overview of the workflow to help expert investigators in detecting anomalous behavior and to pre-select care providers with a profile that could have a higher risk of inappropriate resource use. The health insurance data is first filtered and preprocessed to obtain a single, rich, feature vector for each practitioner. Next, anomaly detection is performed by a number of traditional and state-of-the-art anomaly detection techniques. The model output, for a sample to be investigated, is explained by using SHAP. Finally, the results are summarized in the form of visualisations and summary statistics to the expert investigator. 94

5.2 The preprocessing performed before performing anomaly detection. In close cooperation with experts, the features are filtered and enriched (e.g. using expert knowledge) such that a suitable set of the data and features are extracted that are useful in determining atypical behavior. Numerical features are aggregated per practitioner by their mean, standard deviation, sum, minimum, maximum and mean statistics. The categorical features are embedded using a categorical embedding method and aggregated using the mean. The data is then recombined and standardized by subtracting the mean and dividing by the standard deviation. 95
5.3 The Singular Value Decomposition step of LSA applied to categorical variables. The co-occurrence matrix, relates the categories with the data entries (the practitioner actions), gets decomposed into a rank-k approximation of the category space C, the practitioner actions space P and weights S which relates the two. 96
5.4 The process for explainable anomaly detection. Preprocessed data is provided to n anomaly detection models. The top k anomalies for each models are explained using SHAP. Next, the anomalies are ranked based on how many times they are detected by the anomaly detection methods. finally, The explanations in combination with standard statistics are summarized and provided to an expert to help them in their decision making. . 99
5.5 The SHAP feature contributions for the feature embeddings are summed such that the contribution of the original feature is achieved. For a numerical feature, all values are aggregated per practitioner using the mean, standard deviated, minimum, maximum and median. For a categorical variable, the values are embedded and aggregated by the mean. Both methods expand a feature into multiple dimensions which means there are as many feature contributions (importance scores) per original feature. SHAP has the advantage that these scores can be summed to achieve the overall importance score for the original feature. . . 100

5.6 The SHAP model feature contributions, for a single sample, indicate which feature contribute most to the model output i.e. the anomaly score. The features contributing most to the sample being considered anomalous are then further investigated. For a categorical feature, the distribution of the categorical values are shown and compared to the average of all practitioners. For a numerical feature, the aggregate features (i.e. the minimum, maximum, mean, median and standard deviation) are shown, compared with the mean of the all practitioners and the Z-score is provided. This provides a clear overview for the expert investigator that shows why a practitioner's behavior is flagged as anomalous. 101

A.1 The progress of different GANs on the synthetic ring example. Each column show 3000 samples from the training generator in blue with 3000 samples from the true distribution in green. . . 119
A.2 The progress of different GANs on the synthetic grid example. Each column show 3000 samples from the training generator in blue with 3000 samples from the true distribution in green. . . 120
A.3 Generated images for the Stacked MNIST dataset. Each model is trained with 3 layers and mini-batch size 256. Each square shows 64 samples from the trained generator. 121
A.4 Generated images for CIFAR-10 using a DCGAN architecture. Each square shows 64 samples from the trained generator. . . . 122
A.5 Generated images for CIFAR-100 using a DCGAN architecture. Each square shows 64 samples from the trained generator. . . . 123
A.6 Generated images for STL-10 using a DCGAN architecture. Each square shows 64 samples from the trained generator. 124

B.1 Ring: number of modes (first) and sample quality (second). Grid: number of modes (third) and sample quality (fourth). The x-axis refers to the number of generators of MwuGAN. Sample quality is assessed by counting the number of modes within 3σ of each mode. The blue (resp. red) curve indicates the results obtained for RLS (resp. unif.) sampling with MwuGAN. 136
B.2 Samples per mode for a single GAN trained on the Ring dataset. Only IwMmdGAN, MwuGAN (15), RLS MwuGAN (15), and RLS BuresGAN with a discriminator feature map (Discr.) are capable of covering all the modes. 140
B.3 Samples per mode for a single GAN trained on the Grid dataset. Only the proposed RLS BuresGAN with a discriminator feature map (Discr.) is capable of covering all the modes. 140

C.1 MV-RKM (RBF $k_x(\sigma = 2.1856)$, linear k_y, $s = 144$, lag 70) on the SantaFe chaotic laser dataset showing short-term and long-term predictions in the first and second row respectively. . 146

D.1 The distribution of the number of samples for each practitioner (in log scale) of all practitioners (left) and the known anomalous practitioners(right). 151
D.2 The cardinality of the categorical features for all practitioners (left) and the anomalous practitioners (right). 151
D.3 The distribution of the nomenclature codes for all practitioners (left) and the anomalous practitioners (right). 152

List of Tables

2.1 Experiments on the synthetic data sets. Average (std) over 10 runs. All the models are trained for 25k iterations. 29

2.2 KL-divergence between the generated distribution and true distribution for an architecture with 3 conv. layers for the Stacked MNIST dataset. The number of counted modes assesses mode collapse. The results are obtained after 25k iterations and we report the average(std) over 10 runs. 30

2.3 Stacked MNIST experiment for an architecture with 4 conv. layers. All the models are trained for 25k iterations with a batch size of 64, a learning rate of 2×10^{-4} for Adam and a normal latent distribution. The evaluation is over 10k samples and we report the average(std) over 10 runs. 30

2.4 Generation quality on CIFAR-10, CIFAR-100 and STL-10 with DCGAN architecture. For the CIFAR images, Average(std) over 10 runs and 100k iterations for each. For the STL-10 images, Average(std) over 5 runs and 150k iterations for each. For improving readability, SWD score was multiplied by 100. The symbol '$\sim$' indicates that no meaningful result could be obtained for these parameters. 32

2.5 Best achieved IS and FID, using a ResNet architecture. Results with an asterisk are quoted from their respective papers (std in parenthesis). BuresGAN results were obtained after $300k$ iterations and averaged over 3 runs. The result indicated with † are taken from [233]. For all the methods, the STL-10 images are rescaled to $48 \times 48 \times 3$ in contrast with Table 2.4. 33

2.6 Experiments on the synthetic datasets Ring and Grid. Two RLS BuresGAN are considered: RLSs calculated with the Gaussian kernel (Gauss.) and the next-to-last layer of the discriminator (Discr.). RLS MwuGAN is initialized with RLSs using an implicit feature map associated with the Gaussian kernel. 44

2.7 Experiments on the UNBALANCED 012-MNIST dataset. Two RLS BuresGAN are considered: RLSs calculated with an explicit feature map obtained from a pre-trained classifier (Class.) and the next-to-last layer of the discriminator (Discr.). RLS MwuGAN is initialized with RLSs using the explicit feature maps obtained from a pre-trained classifier. Minority modes are highlighted in black in the first row. 46
2.8 Experiments on the UNBALANCED MNIST dataset. Two RLS BuresGAN variants are considered: RLSs calculated with an explicit feature map obtained from a pre-trained classifier (Class.) and the next-to-last layer of the discriminator (Discr.). RLS MwuGAN is initialized with RLSs using the explicit feature maps obtained from a pre-trained classifier. Only the number of samples in the minority modes are visualized. The number of samples in the remaining modes are given in Supplementary Material. 47
2.9 Experiments on the UNBALANCED 06-CIFAR10 and UNBALANCED 016-CIFAR10 dataset. Including RLS sampling in the StyleGAN2 + Aug. clearly improves the performance. Minority modes are highlighted in black in the second row. 49

3.1 Mean squared error on the forecasted data. Standard deviation for 10 iterations between brackets for the stochastic models. . . 57
3.2 Mean squared error on the forecasted data. Standard deviation for 10 iterations between brackets for the stochastic models. . . . 71

4.1 Left: The categorical variables with their corresponding values of the artificial data set. Right: The corresponding manually created relations present in the data set. 79
4.2 Left: Hierarchical structure of the variables of the nursery data. Right: The Cramer's V score, a measure for association between categorical variables, for the corresponding variables with regards to the final evaluation. 82
4.3 Overview of the classification performance of the baseline one-hot encoding and the categorical embedding methods on the four UCI data sets. The reported values are the average AUC with standard deviation and the best observed AUC for a logistic regression model (LR) and a neural network with one hidden layer (NN). The best performing model is consistently one with as input categorical embeddings. 84

4.4 Overview of the classification performance of the neural embeddings with and without unsupervised pretraining. The reported values are the average AUC with standard deviation and the best observed AUC. For the Breast cancer, CMC and Credit data set the unsupervised methods outperform the neural embeddings without pretraining. In all cases, pretrained neural embeddings with fine-tuning outperform the standard supervised neural embeddings and achieve the best result on all data sets. 84

5.1 The number of samples and features of the data sets used during the experiments. The original data is aggregated and transformed into one-hot encoded and embedding data sets. 102
5.2 Area under the ROC curve of for the one-hot and embedding encoded CM general practitioner data for the following algorithms: Local Outlier Factor (LOF), K Nearest Neighbors (KNN), Isolation Forest (IF), Unsupervised Random Forest (URF), Angle-base Outlier Detection (ABOD), Empirical-Cumulative-distribution-based Outlier Detection (ECOD), Copula-Based Outlier Detection (COPOD), Variational Autoencoder (VAE), Histogram-based Outlier Score (HBOS), Isolation-based Nearest-Neighbor Ensembles (INNE), One-Class Support Vector Machine (OCSVM), Deep Support Vector Data Description (DeepSVDD), Subspace Outlier Detection (SOD), Learnable Unified Neighborhood-based Anomaly Ranking (LUNAR) and Lightweight On-line Detector of Anomalies (LODA). For the stochastic algorithms, the performance shown is the average with standard deviation over 10 different runs. The embedding method offers a clear performance increase for all methods. VAE and LODA achieve the best performance with INNE, OCSVM and COPOD following closely behind. 104

A.1 Average time per iteration in seconds for the convolutional architecture. Average over 5 runs, with std in parenthesis. The batch size is 256. For Stacked MNIST, we use a discriminator architecture with 3 convolutional layers (see Table A.12). For readability, the computation times for the Grid and Ring dataset are multiplied by 10^{-3}. 114
A.2 Inception Score for the best trained models on CIFAR-10, CIFAR-100 and STL-10, with a DCGAN architecture (higher is better). 114
A.3 Comparison for Frobenius norm. Experiments on the synthetic data sets. Average (std) over 10 runs. All the models are trained for 25k iterations. 115

A.4 KL-divergence between the generated distribution and true distribution (Quality, lower is better). The number of counted modes indicates the amount of mode collapse (higher is better). 25k iterations and average and std over 10 runs. Same architecture as in Table 2 in the paper with a discriminator with 2 convolutional layers. The architecture details are in Table A.12. 116
A.5 Ablation study on real image data sets for a convolutional architecture. Frobenius norm is also considered in the place of the Bures metric. Generation quality on CIFAR-10, CIFAR-100 and STL-10 with a DCGAN architecture. Average(std) over 10 runs. 100k iterations for each. For improving readability, SWD score was multiplied by 100. 116
A.6 Ablation study for Stacked MNIST experiment for an architecture with 4 convolutional layers (see Table A.11). All the models are trained for 25k iterations with a batch size of 64, a learning rate of 2e−4 for Adam and a normal latent distribution. The evaluation is over 10k samples and we report an average (std) over 10 runs. 117
A.7 The generator and discriminator architectures for the synthetic examples. 125
A.8 Respectively the MDGAN encoder model and VEEGAN stochastic inverse generator architectures for the synthetic examples. The output of the VEEGAN models are samples drawn from a normal distribution with scale 1 and where the location is learned. 125
A.9 The hyperparameters and architectures of the stacked MNIST experiment performed by the different methods are given here. This represents our best efforts at providing a comparison between the different parameters used. The asterisks ** indicate that the parameters where obtained by the Github repository. Notice that GDPP paper also used 30000 iterations for training DCGAN and unrolled GAN (indicated by *). BuresGAN's column refers to the settings of experiment of Table 2 in the paper and Table A.6 respectively, for which the different values are separated by the symbol &. 126
A.10 The MDGAN encoder model architecture for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not. 126
A.11 The generator and discriminator architectures for the Stacked MNIST experiments with 4 layers for both generator and discrimintor. The BN column indicates whether batch normalization is used after the layer or not. 127

A.12 The generator and discriminator architectures for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not. For the experiments with 2 convolution layers in Table A.4, the final convolution layer is removed in the discriminator. 128
A.13 The CNN architecture of the classifier used during the evaluation of the stackedMNIST experiments. Dropout with a rate of 0.5 is used before the final dense layer. 128
A.14 The generator and discriminator architectures for the CIFAR-10 and CIFAR-100 experiments. The BN column indicates whether batch normalization is used after the layer or not. 129
A.15 The MDGAN encoder model architecture for the CIFAR-10 and CIFAR-100 experiments. The BN column indicates whether batch normalization is used after the layer or not. 129
A.16 The generator and discriminator architectures for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not. 130
A.17 The MDGAN encoder model architecture for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not. 131
A.18 The generator (top) and discriminator (bottom) ResNet architectures for the CIFAR-10 experiments. 132
A.19 The generator (top) and discriminator (bottom) ResNet architectures for the STL-10 experiments. For the experiment with full-sized 96x96 images, an extra upsampling block was added to the generator. 133

B.1 Experiments on the UNBALANCED MNIST dataset. Two variants of RLS BuresGAN are trained. First, RLSs are calculated with an explicit feature map obtained from a pre-trained classifier (Class.). The second version uses the next-to-last layer of the discriminator (Discr.) to compute the RLSs. The RLS MwuGAN is initialized using the RLSs with the explicit feature maps obtained from a pre-trained classifier. 135
B.2 Ablation study over the parameter γ on the synthetic datasets for RLS BuresGAN with a discriminator feature map (Discr.) and Gaussian feature map (Gauss.) with bandwidth $\sigma = 0.15$. No significant difference can be seen from the reported performance as the regularization parameter varies. 136

B.3 Ablation study over the regularization parameter γ and dimension k on the UNBALANCED 012-MNIST dataset for RLS BuresGAN with a discriminator feature map (Discr.) with sketching and an explicit feature map obtained from the next-to-last layer of a pre-trained classifier (Class.) with a UMAP dimensionality reduction and sketched dimensionality reduction. Minority modes are highlighted in black in the first row. The best performance is achieved using the Discr. sketched and Class. UMAP. feature maps with regularization 10^{-4} and dimension $k = 25$. 137
B.4 Ablation study over the regularization parameter γ and dimension k on the UNBALANCED MNIST dataset for RLS BuresGAN with a discriminator feature map (Discr.) with sketching and an explicit feature map obtained from the next-to-last layer of a pre-trained classifier (Class.) with a UMAP dimensionality reduction and sketched dimensionality reduction. Minority modes are highlighted in black in the first row. The best performance is achieved using the Class. UMAP. feature map. 138
B.5 Total training time in seconds, averaged over 10 runs. 139
B.6 The generator (left) and discriminator (right) architectures for the synthetic examples. 142
B.7 The generator (left) and discriminator (right) architectures for the UNBALANCED 012-MNIST and the UNBALANCED MNIST experiments. BN indicates if batch normalization is used. . . . 143
B.8 The CNN architecture of the classifier used during the evaluation of the MNIST experiments. Dropout with a rate of 0.5 is used before the final dense layer. 143

C.1 Datasets used for the experiments. N is the number of training samples, d the input dimension, s the subspace dimension and m the minibatch size. 145

Chapter 1

Introduction

Machine learning is increasingly integrating into our society and has been at the forefront of many recent technological advancements and breakthroughs. In our daily lives, machine learning models provide us with recommendations when shopping online or when browsing social media, help us navigate the road to get to our destination faster, and provide virtual assistants through speech or text interaction.

Artificial intelligence (AI) and machine learning models are a driving force behind certain new scientific breakthroughs and are transforming the way research is performed. In computer science, AI methods assist with programming by generating code based on plain language instructions [103] and are capable of optimizing existing algorithms [118]. In physics, deep learning models can be used to replace or improve computationally intensive simulations, to handle large amounts of noisy data, and to assist researchers with making new discoveries [28, 73, 161]. In medicine, specialized deep learning models help with medical image analysis by detecting, interpreting or locating problems and regions of interest [109]. Electronic health records are another source of data that can be used for machine learning to, for example, assist in predicting readmissions or length of stay [162]. Finally, AlphaFold [188] is a model that helps researchers predict the 3D structure that a protein will take based on a given amino acid sequence. This protein folding problem is important for drug research discovery and for general biological research.

It is clear that machine learning is capable of aiding humans; however, AI models have also achieved superhuman performance in various tasks and domains. As an example, (video) games have shown to be an excellent proving ground to develop and test new machine learning models that can explore and learn in

complicated environments. Models such as AlphaGo and AlphaZero are able to teach themselves how to play games like Chess, Shogi and Go and outperform the top human players and computer programs [192, 190, 191]. In more open environments and real-time games such as Starcraft II and Dota 2, AI systems, AlphaStar [220] and OpenAI Five [142], are able to reach the highest possible level of human play.

Recently, AI services such as ChatGPT [140] and Google Bard [67] have garnered significant interest. These services, which are generally publicly available, can answer questions, generate code, and conduct a dialogue with the user that can seemingly be indistinguishable from another human. These services are backed by large language models, such as GPT-4 [141], LaMDA [203], and LLaMA [212], which are deep neural networks that consist of billions of parameters trained on very large datasets.

One aspect that all these AI models have in common is that they are made possible by the increasing availability and size of data, and the ability to train deeper and larger models thanks to increasing hardware capabilities. However, training such models can be prohibitively expensive. This motivates research into other and better ways to improve the capabilities of machine learning models rather than simply increasing the model size or adding more data. One of such research areas is unsupervised representation learning.

1.1 Unsupervised Representation Learning

Representation learning is a general term that indicates the learning of useful representations, or features, from data that are helpful for a machine learning task. This task can be any machine learning goal or application such as classification, regression, generative modeling, clustering, etc. Representation learning has seen a significant increase in interest due to neural network models. Deep neural networks inherently have powerful representation learning capabilities due to their ability to automatically learn hierarchical features specific to the task they are trained for. However, one has little control over the type of features that neural networks learn. Generally, one wants to guide the representation learning process by preprocessing the data, changing the architecture of the model or by adjusting the training itself, in order to improve the performance of the model or to learn specific types of features.

Often, the performance of machine learning models can be improved by introducing prior knowledge. For example, it is well known that convolutional neural networks work well for image data as their filters across layers work as hierarchical feature detectors [98, 100]. In other cases, it can be beneficial

to impose a specific structure on the learning representations. By way of illustration, one could impose orthogonality or disentanglement for the learned representations in order to improve the explainability of a model [15]. Representation learning is also an integral part of transfer learning. Representations learned for one task could be reused for a different, but related, task. Alternatively, multiple tasks can be considered at once and joint representations can be learned that help solve all problems together [173]. In a similar fashion, representation learning can be employed to multimodal learning where the data consists of different modalities like text, images, tabular data, etc. Typically, specialized techniques are used to learn useful representations of the different modalities which can then be mapped to a joint space [71].

Unsupervised representation learning attempts to automatically discover useful features, relations, or structures from data without the explicit need for labeled data. One of the most popular applications of this type of machine learning is word embeddings in the field of natural language processing. These embeddings are real-valued vector representations of words trained in such a way that they are able to capture useful semantic and syntactic information from text, without the need for labeled data [5]. In fact, the way that large language models are trained, is similar to the training process of the original word embedding techniques. These models are (pre)trained using self-supervision, a framework where a custom task is created that a model has to solve for which the labels are known. For example, in the case of textual data, a word can be removed from a sentence that the model then has to fill in. The advantage of self-supervised methods is that they allow models to be trained on large amounts of data without the need for expensive labels.

1.2 Background

In this section, the methods and general topics, discussed in the thesis, are introduced.

1.2.1 Generative Adversarial Networks

A generative adversarial network (GAN) [65] is a type of generative model that has seen great success in recent years. Generative models learn the data distribution of a given dataset such that the model can later be used to generate new data points in the same input space. Learning a generative model is not an easy task and most methods require approximation techniques when calculating the gradient during training or are difficult to sample from. For

example, restricted Boltzmann machines [180], one of the most well known generative models, is trained by using an approximation method for the gradient, such as contrastive divergence, and is sampled by using Markov chain Monte Carlo. Generative adversarial networks have a number of advantages. It is easy to generate samples from the learned model by sampling from a simple known distribution that is easily mapped to the input space. No approximation methods are required during training as the loss is defined by a second neural network, which is easily differentiable.

A GAN consists of a discriminator $D : \mathbb{R}^d \to \mathbb{R}$ and a generator $G : \mathbb{R}^\ell \to \mathbb{R}^d$ which are typically defined by neural networks and parametrized by real vectors. The tasks of the generator and discriminator are visualized on Fig. 1.1. The generator G maps a point $\boldsymbol{z}$ from a known latent distribution $\mathcal{Z} \in \mathbb{R}^\ell$ to a point in the input space $\mathcal{X} \in \mathbb{R}^d$, often of very high dimension. The latent space $\mathcal{Z}$ can be any distribution that is easy to sample from, however, typically the normal distribution $\mathcal{N}(0, \mathbb{I}_\ell)$ is used. The discriminator D takes as input points from the input space $\mathcal{X}$ such that the value $D(\boldsymbol{x})$ gives the probability that $\boldsymbol{x}$ comes from the empirical distribution. In fact, the discriminator acts as a classifier that attempts to differentiate between samples from the learned distribution and the real data distribution.

The training of a GAN consists in solving

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))], \tag{1.1}$$

by alternating two phases of training. In Eq. (1.1), the expectation in the first term is over the empirical data distribution p_d, while the expectation in the second term is over the generated data distribution p_g, implicitly given by the mapping by G of the latent prior distribution $\mathcal{N}(0, \mathbb{I}_\ell)$. The training of a GAN describes a zero-sum game where the generator wishes to minimize Eq. (1.1) and the discriminator wants to maximize it. The generator wants to fool the discriminator such that it classifies the samples it generates $D(G(\boldsymbol{z}))$ as real. The discriminator wants the opposite, it wants to learn to correctly classify the generated samples as fake and the real samples as belonging to the data distribution. In theory, if the GAN converges, then the discriminator will output a 0.5 for all samples, both real and fake, as both the real and generated distributions are the same and the discriminator is unable to differentiate between them.

In practice, the GAN formulation as a zero-sum game has issues with learning and convergence [65]. Instead, a slightly different loss function is used for the generator as in Eq. (1.3). This loss function is a heuristic, but works exceptionally well in practice and these new, non-saturating, loss functions are used as the basis for all GAN variations.

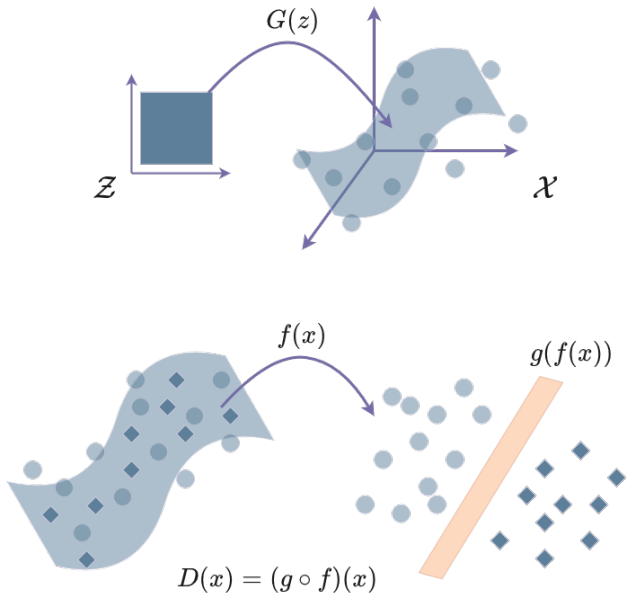


Figure 1.1: The generator G maps samples from the known latent distribution $\mathcal{Z}$ to the input space $\mathcal{X}$. The discriminator D takes as input samples from the input space $\mathcal{X}$ and classifies these samples into fake, generated by G, or real, belonging to the original dataset.

The discriminator loss is still the same and is given by

$$V_D = -\mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))]. \tag{1.2}$$

The novel generator loss now takes the form

$$V_G = -\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\mathbb{I}_\ell)}[\log D(G(\boldsymbol{z}))], \tag{1.3}$$

rather than the second term of Eq. (1.1), for improved training efficiency.

Diversity in Generative Adversarial Networks

We consider two topics that are related to diversity in generative adversarial networks: mode collapse and complete mode coverage.

Mode collapse is one of the most common forms of non-convergence that can occur when training a GAN. This phenomenon manifests itself when the GAN

is unable to learn the complete distribution, and all modes present in the data, and is unable to generate samples with the full diversity present in the dataset. An example of full mode collapse can be observed in Fig. 1.2. In this example, the GAN was trained on the MNIST dataset [99], which consists of handwritten digits 0 to 9. After training, the generator is only able to output samples belonging to a single mode, in this case the digit 1. Full mode collapse, as is the case in this example, is rare; however, a partial form of mode collapse, where the generator generates samples with limited variety, is more common. The exact reason why mode collapse occurs is uncertain; however, it is conjectured that it is a result of the unique training process used in GANs [62]. This can also be explained intuitively. If the generator is able to learn a specific mode in the data especially well then it will be incentivized to continue producing these samples as the main goal of the generator is to fool the discriminator. Furthermore, the occurrence of mode collapse has also been associated with the discriminator's capacity [7, 8]. In fact, if the discriminator is unable to generalize sufficiently well and fails to learn to reject low-diversity samples (e.g. of a single mode), then the generator will never be forced to learn other areas of the data distribution.

Mode collapse is an active research topic, various new GAN types and changes to the original GAN have been proposed in order to prevent mode collapse from occurring. For example, an alternative GAN formulation is that of the Wasserstein GAN [6] where the training objective is obtained through the Kantorovich-Rubinstein duality such that training the GAN corresponds with minimizing the Wasserstein distance between the generated and real distributions. This method allows for the discriminator to be trained until optimality each training step, which prevents mode collapse. A plethora of changes have been proposed to the original GAN formulation as well. UnrolledGAN [125] updates the generator by first updating the discriminator multiple times and unrolling the optimization before calculating the gradients. However, afterwards the discriminator parameters are only updated once and not for all steps in the unrolling process. This allows the generator to be updated by using an optimal, fake discriminator, but prevents overfitting for the actual discriminator. VEEGAN [195] adds an additional reconstructor neural network that maps the real data distribution to the Gaussian input noise. This reconstructor is then trained together with the generator such that the generator and reconstructor become, approximately, the inverses of each other. GDPP-GAN [54] more explicitly attempts to match the diversity of the generated distribution and the real distribution by matching the eigenvectors and eigenvalues of the kernels of the batches in the discriminator feature space. PacGAN [107] employs a packing technique where multiple samples are concatenated together, so a discriminator is better able to gauge the diversity by always considering multiple samples at the same time.

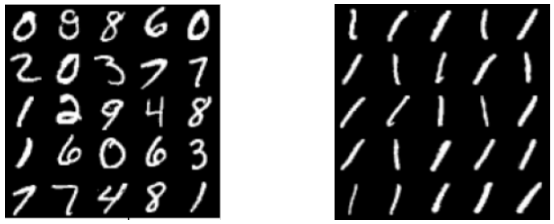


Figure 1.2: An extreme example of mode collapse in GANs. On the left a sample from the original MNIST dataset can be observed while on the right samples generated by the GAN. The generator has learned only a single mode of the dataset corresponding to the number 1.

A related setting to mode collapse is that of complete mode coverage. In this setting, the goal is for the generator to be able to produce samples from all modes in the dataset even when there are few samples available of certain modes. This is especially important when working with biased data. For example, the dataset could be created by a flawed sampling process, or perhaps there is simply not enough data available for a certain mode. In such cases, it may be necessary to correct the present bias in order for the generator to learn the desired distribution. For simple cases, this problem could be tackled from a data perspective, for instance, by employing oversampling methods. However, this requires full knowledge about all possible modes in the dataset and does not take into account the training process of the GAN. More recent solutions change the GAN architecture and the training process itself to achieve full mode coverage. For example, importance weighted generative networks [50] scale the loss function of the GAN for each sample such that the regions where the observed and the target distributions differ, are penalized more. An alternative solution that was proposed is to train multiple generators such that the mixture of generators is able to cover the entire data distribution [243]. The training of the generators happens sequentially, such that for each new trained generator, the data distribution it is trained on is reweighted so that areas that the previous generator performs badly on are given a higher weight.

1.2.2 Restricted Kernel Machines

Restricted Kernel Machines (RKM) [198] is a novel general kernel methods framework characterized by visible and hidden units. RKMs make the connection between Least-Squares Support Vector Machines (LS-SVM) [199] and Restricted Boltzmann Machines (RBM) [180, 78]. More specifically, the kernel PCA formulation in the LS-SVM framework has been transformed into a

representation with visible and hidden units, which is used as the main building block for various RKM models.

Restricted Boltzmann Machines

Restricted Boltzmann machines are undirected graphical models with a bipartite structure consisting of a layer of visible and hidden units. The original formulation of the RBM is defined for binary visible and hidden units, respectively, $\boldsymbol{x}_i \in \mathbb{R}^d$ and $\boldsymbol{h}_j \in \mathbb{R}^s$, which are connected by an interconnection matrix $\boldsymbol{W} \in \mathbb{R}^{d \times s}$ in the energy function:

$$E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{m} -\boldsymbol{x}_i^\top \boldsymbol{W} \boldsymbol{h}_j - \boldsymbol{a}^\top \boldsymbol{x}_i - \boldsymbol{b}^\top \boldsymbol{h}_j, \tag{1.4}$$

where $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}\}$ are the learned parameters of the model. $\boldsymbol{a}$ and $\boldsymbol{b}$ indicate the bias parameters for the visible and hidden units. The energy function can be normalized by a constant known as the partition function:

$$Z = \sum_{\boldsymbol{x}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta})}, \tag{1.5}$$

in order to obtain the joint probability distribution:

$$p(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta}) = \frac{1}{Z} e^{-E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta})}. \tag{1.6}$$

From Eq. (1.5) it can be seen that the probability distribution resulting from the energy function is intractable to evaluate, as it would be necessary to sum over all possible states of $\boldsymbol{x}$ and $\boldsymbol{h}$ [111]. However, due to the bipartite structure, where there are no intralayer connections, it is possible to compute the conditional distributions, $p(\boldsymbol{x} \mid \boldsymbol{h})$ and $p(\boldsymbol{h} \mid \boldsymbol{x})$, in closed form:

$$p(\boldsymbol{x} \mid \boldsymbol{h}; \boldsymbol{\theta}) = \prod_{i=1}^{n} p\left(\boldsymbol{x}_i \mid \boldsymbol{h}; \boldsymbol{\theta}\right) \tag{1.7}$$

$$p(\boldsymbol{h} \mid \boldsymbol{x}; \boldsymbol{\theta}) = \prod_{j=1}^{m} p\left(\boldsymbol{h}_j \mid \boldsymbol{x}; \boldsymbol{\theta}\right), \tag{1.8}$$

where the individual probabilities are given by:

$$p(\boldsymbol{x}_i = 1 \mid \boldsymbol{h}; \boldsymbol{\theta}) = \sigma\left(\sum_{j=1}^{m} \boldsymbol{w}_{ij}\boldsymbol{h}_j + \boldsymbol{a}_i\right) \tag{1.9}$$

and

$$p(\boldsymbol{h}_j = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = \sigma\left(\sum_{i=1}^{n} \boldsymbol{w}_{ij}\boldsymbol{x}_i + \boldsymbol{b}_j\right), \tag{1.10}$$

with σ the sigmoid activation function. Training an RBM is typically performed by approximating the gradient of the log-likelihood [76]:

$$\frac{\partial \log p(\boldsymbol{x})}{\partial w_{ij}} = \langle x_i h_j \rangle_{\text{data}} - \langle x_i h_j \rangle_{\text{model}}, \tag{1.11}$$

where $\langle . \rangle_{\text{data}}$ and $\langle . \rangle_{\text{model}}$ indicate the expectation under the data and model distributions, respectively. Common training algorithms used to train an RBM are contrastive divergence [77] and persistent contrastive divergence [206]. The structure of the RBM allows for efficient calculation of the statistics needed for these learning algorithms. In fact, sampling of the model can be efficiently performed by block gibbs sampling, which alternates calculating $p(\boldsymbol{h} \mid \boldsymbol{x})$ and $p(\boldsymbol{x} \mid \boldsymbol{h})$.

Various extensions have been made to restricted Boltzmann machines. Although the original formulation was defined for binary latent variables, RBMs have been extended for Gaussian latent variables [223]. Additionally, similarly to deep neural network models, deep Boltzmann machines can be created by stacking multiple RBMs such that the latent variables of a layer become the visible variables of the following layer [179].

Least Squares Support Vector Machines formulation for kernel PCA

The LS-SVM kernel PCA formulation for some dataset $X = \{x_i\}_{i=1}^{n}$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ is defined as the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{w}, \boldsymbol{e}_i} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \frac{\gamma}{2}\sum_{i=1}^{n} \boldsymbol{e}_i^2 \\ &\text{s.t. } \boldsymbol{e}_i = \boldsymbol{w}^\top \phi\left(\boldsymbol{x}_i\right), \quad i = 1, \ldots, n. \end{aligned} \tag{1.12}$$

With a centered feature map $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^s$, weight vector $\boldsymbol{w} \in \mathbb{R}^s$ and error variables $e_i \in \mathbb{R}$.

This constrained optimization problem can be solved by constructing the Lagrangian with Lagrange multipliers $\alpha_i \in \mathbb{R}$:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{e}; \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \frac{\gamma}{2}\sum_{i=1}^{n} \boldsymbol{e}_i^2 + \sum^{n} \alpha_i \left(\boldsymbol{e}_i - \boldsymbol{w}^\top \phi\left(\boldsymbol{x}_i\right)\right) \tag{1.13}$$

The optimality conditions are given by:

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w} = \displaystyle\sum_{i=1}^{n} \boldsymbol{\alpha}_i \phi(\boldsymbol{x}_i) & (1.14) \\ \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{e}_i} = 0 \Rightarrow \boldsymbol{\alpha}_i = \gamma \boldsymbol{e}_i & (1.15) \\ \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_i} = 0 \Rightarrow \boldsymbol{e}_i - \boldsymbol{w}^\top \phi\left(\boldsymbol{x}_i\right) = 0. & (1.16) \end{cases}$$

From the optimality conditions, we eliminate w and e and set $\lambda = 1/\gamma$ to get an eigenvalue problem with dual variables as follows:

$$\boldsymbol{K}\boldsymbol{\alpha} = \boldsymbol{\lambda}\boldsymbol{\alpha} \tag{1.17}$$

with $\boldsymbol{K}$ the centered kernel matrix.

Restricted Kernel Machines formulation for kernel PCA

In order to make the link with restricted Boltzmann machines we first reformulate Eq. (1.12) with an interconnection matrix $\boldsymbol{W} \in \mathbb{R}^{d \times s}$:

$$\begin{aligned} &\min_{\boldsymbol{W}, \boldsymbol{e}_i} \frac{1}{2}\operatorname{Tr}(\boldsymbol{W}^\top \boldsymbol{W}) - \frac{\gamma}{2}\sum_{i=1}^{n} \boldsymbol{e}_i^\top \boldsymbol{e}_i \\ &\text{s.t. } \boldsymbol{e}_i = \boldsymbol{W}^\top \phi\left(\boldsymbol{x}_i\right), \quad i = 1, \ldots, n. \end{aligned} \tag{1.18}$$

Next, a formulation with visible and hidden units can be constructed using the conjugate feature duality concept [198]. The latent variables are introduced using the Fenchel-Young inequality [171]:

$$\frac{1}{2\lambda}\boldsymbol{e}^\top \boldsymbol{e} + \frac{\lambda}{2}\boldsymbol{h}^\top \boldsymbol{h} \geq \boldsymbol{e}^\top \boldsymbol{h}, \quad \forall \boldsymbol{e}, \boldsymbol{h} \in \mathbb{R}^s \tag{1.19}$$

Setting $\lambda = 1/\gamma$, using the above property and by eliminating the error variables e_i, Eq. (1.18) can be rewritten in terms of visible and hidden variables as follows:

$$\begin{aligned} J &= \frac{1}{2}\operatorname{Tr}\left(W^\top W\right) - \frac{1}{2\lambda}\sum_{i=1}^{n} \boldsymbol{e}_i^\top \boldsymbol{e}_i \quad \text{s.t. } \boldsymbol{e}_i = W^\top \varphi\left(\boldsymbol{x}_i\right), \forall i \\ &\leq -\sum_{i=1}^{n} \boldsymbol{e}_i^\top \boldsymbol{h}_i + \frac{\lambda}{2}\sum_{i=1}^{n} \boldsymbol{h}_i^\top \boldsymbol{h}_i + \frac{1}{2}\operatorname{Tr}\left(W^\top W\right) \quad \text{s.t. } \boldsymbol{e}_i = W^\top \varphi\left(\boldsymbol{x}_i\right), \forall i \\ &= J_{RKM} = \sum_{i=1}^{n}\left(-\phi\left(\boldsymbol{x}_i\right)^\top W \boldsymbol{h}_i + \frac{\lambda}{2}\boldsymbol{h}_i^\top \boldsymbol{h}_i\right) + \frac{1}{2}\operatorname{Tr}\left(W^\top W\right). \end{aligned} \tag{1.20}$$

This representation is the main objective function of the RKM and is similar to the energy function of an RBM. Specifically, the first term has the same goal where the interconnection matrix $\boldsymbol{W}$ captures the dependencies between the visible units $\boldsymbol{x}_i$ and the latent units $\boldsymbol{h}_i$. The stationary points of the energy function are given by:

$$\begin{cases} \dfrac{\partial J_{RKM}}{\partial \boldsymbol{h_i}} = 0 \implies \boldsymbol{W}^\top \phi\left(\boldsymbol{x}_i\right) = \boldsymbol{\lambda h_i}, \quad \forall i \\ \dfrac{\partial J_{RKM}}{\partial \boldsymbol{W}} = 0 \implies \boldsymbol{W} = \displaystyle\sum_{i=1}^{n} \phi\left(\boldsymbol{x}_i\right) \boldsymbol{h}_i^\top. \end{cases}$$

Eliminating $\boldsymbol{W}$ gives the following eigenvalue problem:

$$\boldsymbol{K}\boldsymbol{H}^\top = \boldsymbol{H}^\top \boldsymbol{\Lambda} \tag{1.21}$$

with $\boldsymbol{H} = \left[\boldsymbol{h}_1 \ldots \boldsymbol{h}_n\right] \in \mathbb{R}^{s \times n}$, $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_s) \in \mathbb{R}^{s \times s}$ where generally $s \leq n$, indicating the number of components and $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathscr{H}}$ are the elements of the kernel matrix $\boldsymbol{K}$ where $\mathscr{H}$ denotes the Reproducing Kernel Hilbert Space. The closed-form solution above is the dual formulation, which means it is not required to explicitly compute $\boldsymbol{W}$. Training an RKM model consists of solving an eigenvalue problem. This is in contrast to restricted Boltzmann machines where the model is trained by approximating the gradient of the log-likelihood of $\boldsymbol{W}$ by using methods such as contrastive divergence [77] and persistent contrastive divergence [206].

The RKM formulation for kernel PCA is the strong representation learning foundation on which RKMs are built and is the building block that is used to

develop more specific algorithms. Furthermore, the representation with visible and hidden units offers more freedom than standard kernel methods when creating models and allows for deep representations similar to neural network architectures [198, 211].

1.2.3 Health Insurance Overconsumption and Fraud

Healthcare is a vital pillar of a modern welfare state. Access to affordable and high-quality healthcare is not only important from a health point of view but is inextricably linked to the economic and social development of a society [22]. For example, a person who cannot work due to illness cannot contribute to the economy. In fact, a well-functioning healthcare system is an investment in the economy and country itself: directly through the creation of health-related jobs and products, and indirectly through the positive impact of a stable and healthy populace on society. This, in turn, contributes to social well-being and stability by taking into account the needs of vulnerable groups by offering equal access to quality care to all [226].

However, healthcare comes at an ever increasing cost. Globally, healthcare expenditure has been steadily increasing, with no end in sight. On Fig. 1.3 this effect can be observed: globally, and for all income groupings of the countries, the percentage of the gross domestic product (GDP) that is spent on healthcare is continuously increasing.

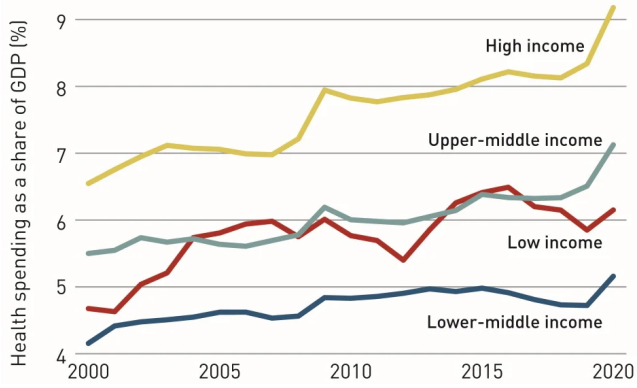


Figure 1.3: Figure taken from the WHO Global Health Expenditure Database [231]. Globally, from low to high income countries, the healthcare spending as a part of their respective GDP has been steadily rising since the year 2000.

For example, the European Union currently spends approximately 1.5 trillion euros annually on healthcare, which on average corresponds to 10.9% of GDP per country in 2020. [57]. In the United States, healthcare spending reached 4.3 trillion dollars [29], which is 18.3% of its GDP. The increased standard of care that is present in modern society means having access to the latest technological advances in medicine and care, which come at a premium [119]. Not to mention large unexpected expenses, for example, as was the case for the recent COVID-19 pandemic, which significantly increased healthcare spending and exacerbated the challenges healthcare systems were already facing [232].

The overall increase in spending is a cause for concern not just economically but also politically, as policy makers are scrambling to deal with increased budget requirements. It is no wonder that all stakeholders involved are attempting to identify areas of improvement such that healthcare spending can be reduced without compromising on the available quality of care. One of such areas is the detection of waste and fraud. In general, waste or fraud is almost never considered on its own, but in the context of a greater whole that contains waste, errors, abuse, fraud and corruption [53]. The main difference between these types of infringements is the intention and the degree of severity. Only in the case of abuse, fraud and corruption can it be said that there is an intent to do wrong by finding loopholes or stretching a rule (abuse) or by actively and intentionally breaking the law (fraud and corruption). In the case of errors, a rule is unintentionally broken. Similar to this, is waste or overconsumption, which consists of care that is unnecessary or needlessly expensive. In such cases, the available medical resources are used inefficiently, which is termed inefficient care. Efficient care refers to a balance between cost, amount and quality of care. When examining the billing data, as available to healthcare insurers, it can be difficult to differentiate between these types of infringements and care providers with valid resource use. Therefore, it is common to group the different categories and look for general anomalies. Alternatively, specific thematic investigations can be performed to examine and search for known types of fraud.

Belgian Mutual Health Insurance

In Belgium, social security includes a compulsory health and disability insurance funded by the social security contributions of its working people. The compulsory health insurance is managed federally by the National Institute for Health and Disability Insurance (abbreviated INAMI or RIZIV in French and Dutch, respectively). This institute allocates funding to the seven accredited mutual health insurance funds [168] which, in turn, reimburse or pay for the healthcare costs of its members. Legally, every person living in Belgium is required to join

a mutual health insurance fund if they are employed or if they are unemployed and receive benefits.

One of the most important tasks of Belgian health insurers is to monitor whether the financial resources in the healthcare sector are spent efficiently. They are, in fact, co-managers of the health insurance system and are a vital part of the enforcement policy concerning fraud, ineffective care and waste in healthcare. Recently, a closer cooperation between the FPS public health, RIZIV and health insurers has been established through a number of action plans concerning "enforcement in healthcare" [167]. An important aspect of the action plans is the commitment to reduce fraud and address the inefficient use of resources in healthcare.

In this context, a scientific project on the detection of healthcare insurance anomalies has been set up at the Christian Health Insurance Fund (CM) [202] under the supervision of the chief medical officer. The data used in the context of this project was only analysed for research purposes, as a proof of concept, without any operational goals. The data subjects were pseudonymised and no re-identification has been performed at any time regardless of whether the samples have been identified as anomalous or not by our anomaly detection techniques. No information of identifiable patients was used during the analyses.

The CM, which has over 4.5 million members, is the largest Belgian mutual health insurer and currently processes over 400 million expenditure records every year. The expenditure records contain a significant amount of information concerning, for example, the actions of a practitioner, medications prescribed for a patient and reimbursements by the CM. In summary, it contains administrative resource use data of healthcare actors in Belgium.

The CM database contains a wealth of information; however, such a large database does not lend itself well to manual checks or investigations. This motivates the use of advanced data-mining and machine learning techniques, in order to automatically process the data and help with detecting anomalous profiles that could indicate waste or fraud. In this thesis, we describe a novel workflow that is able to detect known care providers with atypical behavior and that can help expert investigators make informed decisions regarding possible fraud or overconsumption.

1.3 Research Aims

The work in this thesis consists of three different areas of research: generative adversarial networks, restricted kernel machines and health insurance anomaly

detection. For these topics, we use unsupervised representation learning methods in order to increase performance, tackle biases, improve diversity and explainability of machine learning models.

For the topic of generative adversarial networks, we discuss diversity and bias. When learning the data distribution of a dataset one generally wants to model the complete distribution with all of the different modes or subpopulations present. For example, consider a dataset consisting of medical data of patients with different diseases. As some diseases are rarer than others, the data points related to these rare diseases can be less frequent in the dataset. This can lead to a machine learning model not learning these rare cases sufficiently well or even ignoring them all together, which could have unwanted consequences such as the unequal treatment of patients or diseases. As illustrated by the example with medical data, there are use cases where diversity or bias of data or model can have a large impact and as such are an important research topic in machine learning.

We consider the problems of diversity and bias in GANs in the form of mode collapse and complete mode coverage. A well-known problem with GANs is mode collapse, where the generator is only able to learn, and generate, samples from a small subset of the modes present in the data. A related problem is that of complete mode coverage. In this setting, the goal is to train a generative model that is capable of generating samples from all modes in the data and to reduce the detrimental effect that minority modes have on the model. In this thesis, we propose a number of methods to solve these problems by making use of the rich feature spaces in neural networks, such as those learned by the discriminator.

For the second topic, we consider restricted kernel machines, a novel powerful method with strong representation learning capabilities. Neural network models have certain advantages such as hierarchical, large-scale, and automatic feature learning. However, neural networks are generally regarded as less interpretable than other methods which can limit its applications when explainability is required. Kernel methods, on the other hand, are more interpretable with strong mathematical foundations, but have difficulty with large-scale learning and do not have the advantage of automatic feature learning. Restricted kernel machines attempt to combine the advantages of neural networks and of kernel methods to develop a versatile and unified framework. RKMs have already successfully been used in classification, regression, anomaly detection, generative modeling and kernel principal component analysis. In this thesis, we extend the restricted kernel machines framework towards time-series modeling. Additionally, we show how RKMs can be used to forecast new future data points.

For the third topic, we consider tabular data and health insurance anomaly detection. Tabular data is one of the most common data formats in real-life applications and contains both real and categorical data. However, most machine learning models cannot work with categorical data as input. Typically, categorical data is transformed into a continuous data format using simple methods such as one-hot encoding which do not capture any additional information about the machine learning problem. In this thesis, we show how to transform categorical data into real-valued vectors that are able to capture the relations between the categorical variables and are able to improve the performance of downstream machine learning tasks. Finally, the problem of healthcare insurance anomaly detection is tackled. The overall goal is to create a workflow to help experts in this field to make informed decisions about anomalous profiles of healthcare professionals. This task has a number of unique difficulties due to the nature of the data and the field of healthcare insurance. The data consists of a large number of categorical variables which contain important information about the practitioners and require specialized methods. We experimentally show that our unsupervised categorical embeddings provide a significant increase in the performance of all of the anomaly detection methods used. To be useful, it is not sufficient to be able to identify anomalies; it is also necessary to explain why a certain practitioner was identified as outlying. For explainability, we use Shapley additive explanations (SHAP) in order to explain the output of the models and to help guide the expert investigations.

1.4 Thesis Outline

This section provides an overview of the following chapters of the thesis.

Chapter 2 tackles diversity and bias in generative adversarial networks. In this chapter, we suggest a method for dealing with mode collapse and to achieve complete mode coverage. To address mode collapse, we propose to augment the standard GAN loss function with the Bures distance between the real and generated batches in the feature space of the discriminator. This additional Bures term has as goal to match the real and fake diversity. We show how the Bures distance can conveniently be calculated in the feature space, by way of the covariance matrices, or in the kernel space by employing the kernel matrix of the batches. We show that this novel GAN, which we called BuresGAN, is able to successfully combat mode collapse while at the same time proving to be beneficial to the quality of the generated samples. For the complete mode coverage setting, we propose to employ a diversity sampling method by using ridge leverage scores as weights when sampling the data to create the training batches for the GAN. We propose three different methods to calculate the ridge

leverage scores in feature space. The first is to use an implicit feature mapping in the form of a Gaussian kernel, which works well for simple problems. For more complicated problems, we suggest to use an explicit feature map. For the explicit feature map, we propose to use the next-to-last layer of network pretrained when prior information about the problem is available. Alternatively, the next-to-last layer of the discriminator can be used as the explicit feature map such that the feature map, and thus the sampling, improves together with the GAN. The presented leverage score sampling method improves the overall mode coverage and reduces the detrimental effect that minority modes have on the model.

In **Chapter 3** we consider restricted kernel machines, a novel powerful method with strong representation learning capabilities. RKMs is a framework that has already successfully been used in classification, regression, anomaly detection, generative modeling and kernel principal component analysis. In this chapter, we extend the restricted kernel machines framework towards time-series modeling and forecasting. To do so, we propose two methods: recurrent restricted kernel machines and multi-view kernel principal component analysis. For the first method, we augment the standard RKM objective function by the correlation between the current and past latent variables. This temporal correlation learns a representation and the dynamics of the time-series in latent space. We show that training the model consists of solving a multi-view kernel PCA problem. Forecasting of new, future, data points can be performed by first predicting the next latent variable and then solving a pre-image problem. For the second method, we treat the lagged input and the corresponding future data point as two views of a multi-view kernel PCA problem. The result is a static regression model where future data points are predicted by moving from one view to another. The same model can be used to forecast multiple steps into the future in an autoregressive manner.

In **Chapter 4** we discuss how unsupervised representation learning can be used to create rich vector representations of categorical variables, which can be used for any downstream machine learning task. Drawing inspiration from natural language processing, we show that embedding methods such as CBOW, Skip-gram, GloVe and LSA can be used for categorical data with minimal changes. We show empirically that these methods are able to extract information from the categorical variables themselves and capture the dependencies between the various categorical variables. We illustrate this by visualizing the embeddings using 2D t-SNE and PCA plots where the dependencies between the categorical variables can be clearly seen in the embedding space. Finally, we perform experiments on a number of open-source classification benchmark data sets where we show that models trained with unsupervised categorical embeddings consistently outperform other methods.

In **Chapter 5** we consider the task of creating a workflow for anomaly detection for healthcare insurance data in order to help expert investigators with making informed decisions. Our proposed workflow starts with the preprocessing of the data. For each practitioner in the dataset, time-series data is available with the resource usage of that practitioner for a specific time period. An important difficulty is that the resource usage of a practitioner is largely described by categorical data. We propose to first transform the categorical variables using the method described in Chapter 4. Next, the data is grouped and aggregated, using standard statistics, per practitioners such that a single, rich, vector representation is created which summarizes the behavior of the practitioner. The next step is to perform anomaly detection, for which we perform experiments using a number of state-of-the-art and classical anomaly detection methods in order to determine the most effective techniques. Additionally, the experiments show that the used categorical embeddings show a significant increase in performance for all methods compared to one-hot encoding, which is the standard method in the field. We propose to combine the anomaly scores of multiple anomaly detection models and sort the practitioners based on how anomalous their profiles are. This limits the scope of the search for experts looking for anomalous profiles such that they only need to check the top anomalous practitioners. Additionally, we propose to use Shapley additive explanations (SHAP) to explain the model output of the anomaly detection methods. This provides additional information concerning which features contributed the most to the anomaly score and helps investigators locate the reason for the anomaly. Finally, we discuss how the complete workflow was able to detect a previously disregarded anomalous trend concerning general practitioners.

In **Chapter 6** we summarize the work and provide future research directions based on the work discussed in the thesis.

Chapter 2

Diversity in Generative Adversarial Networks

This chapter has been previously published as:

1. H. De Meulemeester, J. Schreurs, M. Fanuel, B. De Moor, and J. A. Suykens. "The Bures metric for generative adversarial networks". In: *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*. Springer. 2021, pp. 52–66
2. J. Schreurs, H. De Meulemeester, M. Fanuel, B. De Moor, and J. A. Suykens. "Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks". In: *Machine Learning, Optimization, and Data Science: 7th International Conference, LOD 2021, Grasmere, UK, October 4–8, 2021, Revised Selected Papers, Part II*. Springer. 2022, pp. 466–480

Contribution

1. Hannes De Meulemeester developed the method, performed part of the experiments, helped write the article and defined the methodology in collaboration with the coauthors.

2. Hannes De Meulemeester helped define the methodology, performed part of the experiments and wrote the article in collaboration with the coauthors.

2.1 The Bures Metric for Generative Adversarial Networks

Abstract

Generative Adversarial Networks (GANs) are performant generative methods yielding high-quality samples. However, under certain circumstances, the training of GANs can lead to mode collapse or mode dropping. To address this problem, we use the last layer of the discriminator as a feature map to study the distribution of the real and the fake data. During training, we propose to match the real batch diversity to the fake batch diversity by using the Bures distance between covariance matrices in this feature space. The computation of the Bures distance can be conveniently done in either feature space or kernel space in terms of the covariance and kernel matrix respectively. We observe that diversity matching reduces mode collapse substantially and has a positive effect on sample quality. On the practical side, a very simple training procedure is proposed and assessed on several data sets.

2.1.1 Introduction

In several machine learning applications, data is assumed to be sampled from an implicit probability distribution. The estimation of this empirical implicit distribution is often intractable, especially in high dimensions. To tackle this issue, generative models are trained to provide an algorithmic procedure for sampling from this unknown distribution. Popular approaches are Variational Auto-Encoders proposed in [90], Generating Flow models [166] and Generative Adversarial Networks (GANs) [63]. The latter are particularly successful approaches to produce high quality samples, especially in the case of natural images, though their training is notoriously difficult. The vanilla GAN consists of two networks: a generator and a discriminator. The generator maps random noise, usually drawn from a multivariate normal, to fake data in input space. The discriminator estimates the likelihood ratio of the generator network to the data distribution. It often happens that a GAN generates samples only from a few of the many modes of the distribution. This phenomenon is called 'mode collapse'.

Contribution. We propose BuresGAN: a generative adversarial network which has the objective function of a vanilla GAN complemented by an additional term, given by the squared Bures distance between the covariance matrix of

real and fake batches in a latent space. This loss function promotes a matching of fake and real data in a feature space $\mathbb{R}^f$, so that mode collapse is reduced. Conveniently, the Bures distance also admits both a feature space and kernel based expression. Contrary to other related approaches such as in [32] or [195], the architecture of the GAN is unchanged, only the objective is modified. We empirically show that the proposed method is robust when it comes to the choice of architecture and does not require an additional fine architecture search. Finally, an extra asset of BuresGAN is that it yields competitive or improved IS and FID scores compared with the state of the art on CIFAR-10 and STL-10 using a ResNet architecture.

Related works. The Bures distance is closely related to the Fréchet distance [51] which is a 2-Wasserstein distance between multivariate normal distributions. Namely, the Fréchet distance between multivariate normals of equal means is the Bures distance between their covariance matrices. The Bures distance is also equivalent to the exact expression for the 2-Wasserstein distance between two elliptically contoured distributions with the same mean as shown in [60] and [159]. Noticeably, it is also related to the Fréchet Inception Distance score (FID), which is a popular manner to assess the quality of generative models. This score uses the Fréchet distance between real and generated samples in the feature space of a pre-trained inception network as it is explained in [181] and [75].

There exist numerous works aiming to improve training efficiency of generative networks. For mode collapse evaluation, we compare BuresGAN to the most closely related works. GDPP-GAN [54] and VEEGAN [195] also try to enforce diversity in 'latent' space. GDPP-GAN matches the eigenvectors and eigenvalues of the real and fake diversity kernel. In VEEGAN, an additional reconstructor network is introduced to map the true data distribution to Gaussian random noise. In a similar way, architectures with two discriminators are analysed by [136], while MADGAN [61] uses multiple discriminators and generators. A different approach is taken by Unrolled-GAN [125] which updates the generator with respect to the unrolled optimization of the discriminator. This allows training to be adjusted between using the optimal discriminator in the generator's objective, which is ideal but infeasible in practice. Wasserstein GANs [6, 69] leverage the 1-Wasserstein distance to match the real and generated data distributions. In MDGAN [32], a regularization is added to the objective function, so that the generator can take advantage of another similarity metric with a more predictable behavior. This idea is combined with a penalization of the missing modes. Some other recent approaches to reducing mode collapse are variations of WGAN [233]. Entropic regularization has been also proposed in PresGAN [49], while metric embeddings were used in the paper introducing

BourGAN [234]. A simple packing procedure which significantly reduces mode collapse was proposed in PacGAN [107] that we also consider hereafter in our comparisons.

2.1.2 Method

A GAN consists of a discriminator $D : \mathbb{R}^d \to \mathbb{R}$ and a generator $G : \mathbb{R}^\ell \to \mathbb{R}^d$ which are typically defined by neural networks and parametrized by real vectors. The value $D(\boldsymbol{x})$ gives the probability that $\boldsymbol{x}$ comes from the empirical distribution, while the generator G maps a point $\boldsymbol{z}$ in the latent space $\mathbb{R}^\ell$ to a point in the input space $\mathbb{R}^d$. The training of a GAN consists in solving

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))], \tag{2.1}$$

by alternating two phases of training. In (2.1), the expectation in the first term is over the empirical data distribution p_d, while the expectation in the second term is over the generated data distribution p_g, implicitly given by the mapping by G of the latent prior distribution $\mathcal{N}(0, \mathbb{I}_\ell)$. It is common to define and minimize the discriminator loss by

$$V_D = -\mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))]. \tag{2.2}$$

In practice, it is proposed in [63] to minimize generator loss

$$V_G = -\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\mathbb{I}_\ell)}[\log D(G(\boldsymbol{z}))], \tag{2.3}$$

rather than the second term of (2.1), for an improved training efficiency.

Matching real and fake data covariance. To prevent mode collapse, we encourage the generator to sample fake data of similar diversity to real data. This is achieved by matching the sample covariance matrices of real and fake data respectively. Covariance matching and similar ideas were explored for GANs in [133] and [54]. In order to compare covariance matrices, we propose to use the squared Bures distance between positive semi-definite $\ell \times \ell$ matrices[1] [18], i.e.,

$$\begin{aligned}\mathcal{B}(A, B)^2 &= \min_{U\in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 \\ &= \mathrm{Tr}(A + B - 2(A^{\frac{1}{2}}BA^{\frac{1}{2}})^{\frac{1}{2}}).\end{aligned}$$

[1]The Bures distance between two PSD matrices A and B may additionally be defined as a minimization problem over $U \in O(\ell)$ with $O(\ell)$ the set of $\ell \times \ell$ orthonormal matrices

Being a Riemannian metric on the manifold of positive semi-definite matrices [121], the Bures metric is adequate to compare covariance matrices. The covariances are defined in a feature space associated to the discriminator. Namely, let $D(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}))$, where $\boldsymbol{w}$ is the weight vector of the last dense layer and σ is the sigmoid function. The last layer of the discriminator, denoted by $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{d_\phi}$, naturally defines a feature map. We use the normalization $\bar{\boldsymbol{\phi}}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})/\|\boldsymbol{\phi}(\boldsymbol{x})\|_2$, after the centering of $\boldsymbol{\phi}(\boldsymbol{x})$. Then, we define a covariance matrix as follows: $C(p) = \mathbb{E}_{\boldsymbol{x}\sim p}[\bar{\boldsymbol{\phi}}(\boldsymbol{x})\bar{\boldsymbol{\phi}}(\boldsymbol{x})^\top]$. For simplicity, we denote the real data and generated data covariance matrices by $C_d = C(p_d)$ and $C_g = C(p_g)$, respectively. Our proposal is to replace the generator loss by $V_G + \lambda \mathcal{B}(C_d, C_g)^2$. The value $\lambda = 1$ was found to yield good results in the studied data sets. Two specific training algorithms are proposed. Algorithm 2.1 deals with the squared Bures distance as an additive term to the generator loss, while an alternating training is discussed in Appendix A.2.1 and does not introduce an extra parameter. The training described in Algorithm 2.1 is analogous to

Algorithm 2.1: BuresGAN

Sample a real and fake batch ;
Update G by minimizing $V_G + \lambda \mathcal{B}(\hat{\boldsymbol{C}}_d, \hat{\boldsymbol{C}}_g)^2$;
Update D by maximizing $-V_D$;

the training of GDPP GAN, although the additional generator loss is rather different. The computational advantage of the Bures distance is that it admits two expressions which can be evaluated numerically in a stable way. Namely, there is no need to calculate a gradient update through an eigendecomposition.

Feature space expression. In the training procedure, real $\boldsymbol{x}_i^{(d)}$ and fake data $\boldsymbol{x}_i^{(g)}$ with $1 \leq i \leq b$ are sampled respectively from the empirical distribution and the mapping of the normal distribution $\mathcal{N}(0, \mathbb{I}_\ell)$ by the generator. Consider the case where the batch size b is larger than the feature space dimension. Let the embedding of the batches in feature space be $\Phi_\alpha = [\boldsymbol{\phi}(\boldsymbol{x}_1^{(\alpha)}), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_b^{(\alpha)})]^\top \in \mathbb{R}^{b\times d_\phi}$ with $\alpha = d, g$. The covariance matrix of one batch in feature space[2] is $\hat{C} = \bar{\Phi}^\top \bar{\Phi}$, where $\bar{\Phi}$ is the ℓ_2-normalized centered feature map of the batch. Numerical instabilities can be avoided by adding a small number, e.g. 1e$-$14, to the diagonal elements of the covariance matrices, so that, in practice, we only deal with strictly positive definite matrices. From the computational perspective, an interesting alternative expression for the Bures distance is given by

$$\mathcal{B}\left(C_d, C_g\right)^2 = \mathrm{Tr}\big(C_d + C_g - 2(C_g C_d)^{\frac{1}{2}}\big), \tag{2.4}$$

[2]For simplicity, we omit the normalization by $\frac{1}{b-1}$ in front of the covariance matrix.

whose computation requires only one matrix square root. This identity can be obtained from Lemma 2.1.1. Note that an analogous result is proved in [138].

Lemma 2.1.1. *Let A and B be symmetric positive semidefinite matrices of the same size and let $B = Y^\top Y$. Then, we have: (i) AB is diagonalizable with nonnegative eigenvalues, and (ii)* $\mathrm{Tr}((AB)^{\frac{1}{2}}) = \mathrm{Tr}((YAY^\top)^{\frac{1}{2}})$.

Proof. (i) is a consequence of Corollary 2.3 in [80]. (ii) We now follow [138]. Thanks to (i), we have $AB = PDP^{-1}$ where D is a nonnegative diagonal and the columns of P contain the right eigenvectors of AB. Therefore, $\mathrm{Tr}((AB)^{1/2}) = \mathrm{Tr}(D^{1/2})$. Then, $YAY^\top$ is clearly diagonalizable. Let us show that it shares its nonzero eigenvalues with AB. a) We have $ABP = PD$, so that, by multiplying of the left by Y, it holds that $(YAY^\top)YP = YPD$. b) Similarly, suppose that we have the eigenvalue decomposition $YAY^\top Q = Q\Lambda$. Then, we have $BAY^\top Q = Y^\top Q\Lambda$ with $B = Y^\top Y$. This means that the non-zero eigenvalues of $YAY^\top$ are also eigenvalues of BA. Since A and B are symmetric, this completes the proof. □

Kernel based expression. Alternatively, if the feature space dimension f is larger than the batch size b, it is more efficient to compute $\mathcal{B}(\hat{C}_d, \hat{C}_g)$ thanks to $b \times b$ kernel matrices: $K_d = \bar{\Phi}_d \bar{\Phi}_d^\top$, $K_g = \bar{\Phi}_g \bar{\Phi}_g^\top$ and $K_{dg} = \bar{\Phi}_d \bar{\Phi}_g^\top$. Then, we have the kernel based expression

$$\mathcal{B}(\hat{C}_d, \hat{C}_g)^2 = \mathrm{Tr}\left(K_d + K_g - 2\left(K_{dg}K_{dg}^\top\right)^{\frac{1}{2}}\right), \tag{2.5}$$

which allows to calculate the Bures distance between covariance matrices by computing a matrix square root of a $b \times b$ matrix. This is a consequence of Lemma 2.1.2.

Lemma 2.1.2. *The matrices $X^\top XY^\top Y$ and $YX^\top XY^\top$ are diagonalizable with nonnegative eigenvalues and share the same non-zero eigenvalues.*

Proof. The result follows from Lemma 1 and its proof, where $A = X^\top X$ and $B = Y^\top Y$. □

Connection with Wasserstein GAN and integral probability metrics. The Bures distance is proportional to the 2-Wasserstein distance $\mathcal{W}_2$ between two ellipically contoured distributions with the same mean [60]. For instance, in the case of multivariate normal distributions, we have

$$\mathcal{B}(A, B)^2 = \min_{\pi} \mathbb{E}_{(X,Y)\sim\pi} \|X - Y\|_2^2 \text{ s.t. } \begin{cases} X \sim \mathcal{N}(0, A) \\ Y \sim \mathcal{N}(0, B), \end{cases}$$

where the minimization is over the joint distributions π. More precisely, in this paper, we make the approximation that the implicit distribution of the real and generated data in the feature space $\mathbb{R}^{d_\phi}$ (associated to $\boldsymbol{\phi}(\boldsymbol{x})$) are elliptically contoured with the same mean. Under different assumptions, the Generative Moment Matching Networks [165, 102] work in the same spirit, but use a different approach to match covariance matrices. On the contrary, WGAN uses the Kantorovich dual formula for the 1-Wasserstein distance: $\mathcal{W}_1(\alpha, \beta) = \sup_{f \in \mathrm{Lip}} \int f \, \mathrm{d}(\alpha - \beta)$, where α, β are signed measures. Generalizations of such integral formulae are called integral probability metrics (see for instance [19]). Here, f is the discriminator, so that the maximization over Lipschitz functions f plays the role of the maximization over discriminator parameters in the min-max game of (2.1). Then, in the training procedure, this maximization alternates with a minimization over the generator parameters.

We can now discuss the connection with Wasserstein GAN. Coming back to the definition of BuresGAN, we can now explain that the 2-Wasserstein distance provides an upper bound on an integral probability metric. Then, if we assume that the densities are elliptically contoured distributions in feature space, the use of the Bures distance to calculate $\mathcal{W}_2$ allows to spare the maximization over the discriminator parameters – and this motivates why the optimization of $\mathcal{B}$ only influences updates of the generator in Algorithm 2.1. Going more into detail, the 2-Wasserstein distance between two probability densities (w.r.t. the same measure) is equivalent to a Sobolev dual norm, which can be interpreted as an integral probability metric. Indeed, let the Sobolev semi-norm $\|f\|_{H^1} = (\int \|\nabla f(x)\|^2 \, \mathrm{d}x)^{1/2}$. Then, its dual norm over signed measures is defined as $\|\nu\|_{H^{-1}} = \sup_{\|f\|_{H^1} \leq 1} \int f \, \mathrm{d}\nu$. It is then shown in [158] and [159] that there exist two positive constants c_1 and c_2 such that

$$c_1 \|\alpha - \beta\|_{H^{-1}} \leq \mathcal{W}_2(\alpha, \beta) \leq c_2 \|\alpha - \beta\|_{H^{-1}}.$$

Hence, the 2-Wasserstein distance gives an upper bound on an integral probability metric.

Algorithmic details. The matrix square root in (2.4) and (2.5) is obtained thanks to the Newton-Schultz algorithm which is inversion free and can be efficiently calculated on GPUs since it involves only matrix products. In practice, we found 15 iterations of this algorithm to be sufficient for the small scale data sets, while 20 iterations were used for the ResNet examples. A small regularization term 1e$-$14 is added to the matrix diagonal for stability. The latent prior distribution is $\mathcal{N}(0, \mathbb{I}_\ell)$ with $\ell = 100$ and the parameter in Algorithm 2.1 is always set to $\lambda = 1$. In the tables hereafter, we indicate the largest scores in bold, although we invite the reader to also consider the standard deviation.

2.1.3 Empirical Evaluation of Mode Collapse

BuresGAN's performances on synthetic data, artificial and real images are compared with the standard DCGAN [181], WGAN-GP, MDGAN, Unrolled GAN, VEEGAN, GDPP and PacGAN. We want to emphasize that the purpose of this experiment is not to challenge these baselines, but to report the improvement obtained by adding the Bures metric to the objective function. It would be straightforward to add the Bures loss to other GAN variants, as well as most GAN architectures, and we would expect an improvement in mode coverage and generation quality. In the experiments, we notice that adding the Bures loss to the vanilla GAN already significantly improves the results.

A low dimensional feature space ($d_\phi = 128$) is used for the synthetic data so that the feature space formula in (2.4) is used, while the dual formula in (2.5) is used for the image data sets (Stacked MNIST, CIFAR-10, CIFAR-100 and STL-10) for which the feature space is larger than the batch size. The architectures used for the image data sets are based on DCGAN [160], while results using ResNets are given in Section 2.1.5. All images are scaled in between -1 and 1 before running the algorithms. Additional information on the architectures and data sets is given in SM. The hyperparameters of other methods are typically chosen as suggested in the authors' reference implementation. The number of unrolling steps in Unrolled GAN is chosen to be 5. For MDGAN, both versions are implemented but only MDGAN-v2 gives interesting results. The first version, which corresponds to the mode regularizer, has hyperparameters $\lambda_1 = 0.2$ and $\lambda_2 = 0.4$, for the second version, which corresponds to manifold diffusion training for regularized GANs, has $\lambda = 10^{-2}$. WGAN-GP uses $\lambda = 10.0$ and $n_{\text{critic}} = 5$. All models are trained using Adam [88] with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and learning rate 10^{-3} for both the generator and discriminator. Unless stated otherwise, the batch size is 256. Examples of random generations of all the GANs are given in SM. Notice that in this section we report the results achieved only at the end of the training.

Artificial Data

Synthetic. RING is a mixture of eight two-dimensional isotropic Gaussians in the plane with means $2.5 \times (\cos((2\pi/8)i), \sin((2\pi/8)i))$ and std 0.01 for $1 \leq i \leq 8$. GRID is a mixture of 25 two-dimensional isotropic normals in the plane with means separated by 2 and with standard deviation 0.05. All models have the same architecture, with $\ell = 256$ following [54], and are trained for 25k iterations. The evaluation is done by sampling 3k points from the generator network. A sample is counted as high quality if it is within 3 standard deviations

of the nearest mode. The experiments are repeated 10 times for all models and their performance is compared in Table 2.1.

BuresGAN consistently captures all the modes and produces the highest quality samples. The training progress of the BuresGAN is shown on Figure 2.1, where we observe that all the modes early on in the training procedure, afterwards improving the quality. The training progress of the other GAN models listed in Table 2.1 is given in Appendix A.3.1. Although BuresGAN training times are larger than most other methods for this low dimensional example, we show in Appendix A.1 that BuresGAN scales better with the input data dimension and architecture complexity.

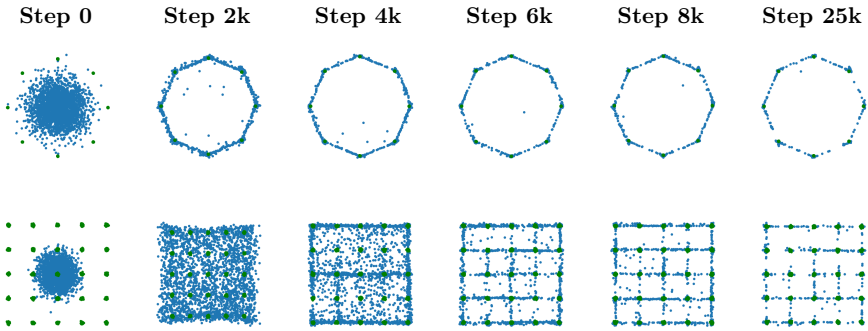


Figure 2.1: Figure accompanying Table 2.1, the progress of BuresGAN on the synthetic examples. Each column shows 3k samples from the training of the generator in blue and 3k samples from the true distribution in green.

Stacked MNIST. The Stacked MNIST data set is specifically constructed to contain 1000 known modes. This is done by stacking three digits, sampled uniformly at random from the original MNIST data set, each in a different channel. BuresGAN is compared to the other methods and are trained for 25k iterations. For the performance evaluation, we follow [125] and use the following metrics: the number of captured modes measures mode collapse and the KL divergence, which also measures sample quality. The mode of each generated image is identified by using a standard MNIST classifier which is trained up to 98.43% accuracy on the validation set (see appendix A.13), and classifies each channel of the fake sample. The same classifier is used to count the number of captured modes. The metrics are calculated based on 10k generated images for all the models. Generated samples from BuresGAN are given in Figure 2.2. As it was observed by previous papers such as in [107] and [234], even the vanilla GAN can achieve an excellent mode coverage for certain architectures.

Table 2.1: Experiments on the synthetic data sets. Average (std) over 10 runs. All the models are trained for 25k iterations.

	Grid (25 modes)		Ring (8 modes)	
	Nb modes	% in 3σ	Nb modes	% in 3σ
GAN	22.9(4)	76(13)	7.4(2)	76(25)
WGAN-GP	24.9(0.3)	77(10)	7.1(1)	9(5)
MDGAN-v2	**25**(0)	68(11)	5(3)	20(15)
Unrolled	19.7(1)	78(19)	**8**(0)	77(18)
VEEGAN	**25**(0)	67(3)	**8**(0)	29(5)
GDPP	20.5(5)	79(23)	7.5(0.8)	73(25)
PacGAN2	23.6(4)	65(28)	**8**(0)	81(15)
BuresGAN (ours)	**25**(0)	**82**(1)	**8**(0)	**82**(4)

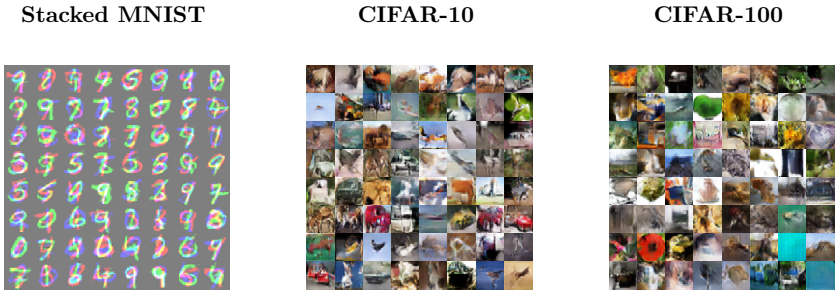


Figure 2.2: Generated samples from a trained BuresGAN, with a DCGAN architecture.

To study mode collapse on this data set, we performed extensive experiments for multiple discriminator layers and multiple batch sizes. In the main section, the results for a 3 layer discriminator are reported in Table 2.2. Additional experiments can be found in supplementary material. An analogous experiment as in VEEGAN [195] with a challenging architecture including 4 convolutional layers for both the generator and discriminator was also included for completeness; see Table 2.3. Since different authors, such as in the PacGAN's paper, use slightly different setups, we also report in SM the specifications of the different settings. We want to emphasize the interests of this simulation is to compare GANs in the same consistent setting, while the results may vary from those reported in e.g. in [195] since some details might differ.

Table 2.2: KL-divergence between the generated distribution and true distribution for an architecture with 3 conv. layers for the Stacked MNIST dataset. The number of counted modes assesses mode collapse. The results are obtained after 25k iterations and we report the average(std) over 10 runs.

		Nb modes(↑)			KL div.(↓)		
	Batch size	64	128	256	64	128	256
3 conv. layers	GAN	993.3(3.1)	995.4(1.7)	**998.3**(1.2)	0.28(0.02)	**0.24**(0.02)	0.21(0.02)
	WGAN-GP	980.2(57)	838.3(219)	785.1(389)	**0.26**(0.34)	1.05(1)	1.6(2.4)
	MDGAN-v1	233.8(250)	204.0(202)	215.5(213)	5.0(1.6)	4.9(1.3)	5.0(1.2)
	MDGAN-v2	299.9(457)	300.4(457)	200.0(398)	4.8(3.0)	4.7(3.0)	5.5(2.6)
	UnrolledGAN	934.7(107)	874.1(290)	884.9(290)	0.72(0.51)	0.98(1.46)	0.90(1.4)
	VEEGAN	974.2(10.3)	687.9(447)	395.6(466)	0.33(0.05)	2.04(2.61)	3.52(2.64)
	GDPP	894.2(298)	897.1(299)	997.5(1.4)	0.92(1.92)	0.88(1.93)	**0.20**(0.02)
	PacGAN2	989.8(4.0)	993.3(4.8)	897.7(299)	0.33(0.02)	0.29(0.04)	0.87(1.94)
	BuresGAN (ours)	**993.5**(2.7)	**996.3**(1.6)	997.1(2.4)	0.29(0.02)	0.25(0.02)	0.23(0.01)

Table 2.3: Stacked MNIST experiment for an architecture with 4 conv. layers. All the models are trained for 25k iterations with a batch size of 64, a learning rate of 2×10^{-4} for Adam and a normal latent distribution. The evaluation is over 10k samples and we report the average(std) over 10 runs.

		Nb modes(↑)	KL div.(↓)
4 conv. layers	GAN	21.6(25.8)	5.10(0.83)
	WGAN-GP	**999.7**(0.6)	**0.11**(0.006)
	MDGAN-v2	729.5(297.9)	1.76(1.65)
	UnrolledGAN	24.3(23.61)	4.96(0.68)
	VEEGAN	816.1(269.6)	1.33(1.46)
	GDPP	33.3(39.4)	4.92(0.80)
	PacGAN2	972.4(12.0)	0.45(0.06)
	BuresGAN (ours)	989.9(4.7)	0.38(0.06)

Interestingly, for most models, an improvement is observed in the quality of the images – KL divergence – and in terms of mode collapse – number of modes attained – as the size of the batch increases. For the same batch size, architecture and iterations, the image quality is improved by BuresGAN, which consistently performs well over all settings. The other methods show a higher variability over the different experiments. MDGANv2, VEEGAN, GDPP and WGAN-GP often have an excellent single run performance. However, when increasing the number of discriminator layers, the training of these models has

a tendency to collapse more often as indicated by the large standard deviation. Vanilla GAN is one of the best performing models in the variant with 3 layers.

We observe in Table 2.3 for the additional experiment that vanilla GAN and GDPP collapse for this architecture. WGAN-GP yields the best result and is followed by BuresGAN. However, as indicated in Table 2.2, WGAN-GP is sensitive to the choice of architecture and hyperparameters and its training time is also longer as it can be seen from the corresponding timings table in supplementary material. More generally, these results depend heavily on the precise architecture choice and to a lesser extent on the batch size. These experiments further confirm the finding that most GAN models, including the standard version, can learn all modes with careful and sufficient architecture tuning [107, 124]. Finally, it can be concluded that BuresGAN performs well for all settings, showing that it is robust when it comes to batch size and architecture.

2.1.4 Real Images

Metrics. Image quality is assessed thanks to the Inception Score (IS), Fréchet Inception Distance (FID) and Sliced Wasserstein Distance (SWD). The latter was also used in [54] and [85] to evaluate image quality as well as mode-collapse. In a word, SWD evaluates the multiscale statistical similarity between distributions of local image patches drawn from Laplacian pyramids. A small Wasserstein distance indicates that the distribution of the patches is similar, thus real and fake images appear similar in both appearance and variation at this spatial resolution. The metrics are calculated based on 10k generated images for all the models.

CIFAR data sets. We evaluate the GANs on the $32\times32\times3$ CIFAR data sets, for which all models are trained for 100k iterations with a convolutional architecture. In Table 2.4, the best performance is observed for BuresGAN in terms of image quality, measured by FID and Inception Score, and in terms of mode collapse, measured by SWD. We also notice that UnrolledGAN, VEEGAN and WGAN-GP have difficulty converging to a satisfactory result for this architecture. This is contrast to the 'simpler' synthetic data and the Stacked MNIST data set, where the models attain a performance comparable to BuresGAN. Also, for this architecture and number of training iterations, MDGAN did not converge to a meaningful result in our simulations. In [6], WGAN-GP achieves a very good performance on CIFAR-10 with a ResNet architecture which is considerably more complicated than the DCGAN used here. Therefore, results with a Resnet architecture are reported in Section 2.1.5.

Table 2.4: Generation quality on CIFAR-10, CIFAR-100 and STL-10 with DCGAN architecture. For the CIFAR images, Average(std) over 10 runs and 100k iterations for each. For the STL-10 images, Average(std) over 5 runs and 150k iterations for each. For improving readability, SWD score was multiplied by 100. The symbol '∼' indicates that no meaningful result could be obtained for these parameters.

	CIFAR-10			CIFAR-100			STL-10		
	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)
GAN	5.67(0.22)	59(8.5)	3.7(0.9)	5.2(1.1)	91.7(66)	7.8(4.9)	2.9(1.8)	237(54)	12.3(4.1)
WGAN-GP	2.01(0.47)	291(87)	8.3(1.9)	1.2(0.5)	283(113)	9.7(2.5)	∼	∼	∼
UnrolledGAN	3.1(0.6)	148(42)	9.0(5)	3.2(0.7)	172.9(40)	13.1(9.2)	∼	∼	∼
VEEGAN	2.5(0.6)	198(33.5)	12.0(3)	2.8(0.7)	177.2(27)	12.8(3.9)	∼	∼	∼
GDPP	5.76(0.27)	62.1(5.5)	4.1(1.1)	5.9(0.2)	65.0(8)	4.4(1.9)	3.3(2.2)	232(84)	8.2(4.0)
PacGAN2	5.51(0.18)	60.61(5.9)	2.95(1)	5.6(0.1)	59.9(5.2)	4.0(1.8)	4.7(1.5)	161(36)	8.1(4.3)
BuresGAN (ours)	**6.34**(0.17)	**43.7**(0.9)	**2.1**(0.6)	**6.5**(0.1)	**47.2**(1.2)	**2.1**(1.0)	**7.6**(0.3)	**109**(7)	**2.3**(0.3)

CIFAR-100 data set consists of 100 different classes and is therefore more diverse. Compared to the original CIFAR-10 data set, the performance of the studied GANs remains almost the same. An exception is vanilla GAN, which shows a higher presence of mode collapse as measured by SWD.



Figure 2.3: Images generated by BuresGAN with a ResNet architecture for CIFAR-10 (left) and STL-10 (right) data sets. The STL-10 samples are full-sized 96×96 images.

STL-10. The STL-10 data set includes higher resolution images of size $96 \times 96 \times 3$. The best performing models from previous experiments are trained for 150k iterations. Samples of generated images from BuresGAN are given on Figure 2.3. The metrics are calculated based on 5k generated images for all

the models. Compared to the previous data sets, GDPP and vanilla GAN are rarely able to generate high quality images on the higher resolution STL-10 data set. Only BuresGANs are capable of consistently generating high quality images as well as preventing mode collapse, for the same architecture.

Timings. The computing times for these data sets are in supplementary material. For the same number of iterations, BuresGAN training time is comparable to WGAN-GP training for the simple data in Table 2.1. For more complicated architectures, BuresGAN scales better and the training time was observed to be significantly shorter with respect to WGAN-GP and several other methods.

2.1.5 High Quality Generation using a ResNet Architecture

Table 2.5: Best achieved IS and FID, using a ResNet architecture. Results with an asterisk are quoted from their respective papers (std in parenthesis). BuresGAN results were obtained after $300k$ iterations and averaged over 3 runs. The result indicated with † are taken from [233]. For all the methods, the STL-10 images are rescaled to $48 \times 48 \times 3$ in contrast with Table 2.4.

	CIFAR-10		STL-10	
	IS (↑)	FID (↓)	IS (↑)	FID (↓)
WGAN-GP ResNet [69]*	7.86(0.07)	18.8†	/	/
InfoMax-GAN [101]*	8.08(0.08)	17.14(0.20)	8.54(0.12)	37.49(0.05)
SN-GAN ResNet [132]*	8.22(0.05)	21.7(0.21)	9.10(0.04)	40.1(0.50)
ProgressiveGAN [85]*	8.80(0.05)	/	/	/
CR-GAN [240]*	8.4	14.56	/	/
NCSN [194]*	**8.87**(0.12)	25.32	/	/
Improving MMD GAN [221]*	8.29	16.21	9.34	37.63
WGAN-div [233]*	/	18.1†	/	/
BuresGAN ResNet (Ours)	8.81(0.08)	**12.91**(0.40)	**9.67**(0.19)	**31.42**(1.01)

As noted by [112], a fair comparison should involve GANs with the same architecture, and this is why we restricted in our paper to a classical DCGAN architecture. It is natural to question the performance of BuresGAN with a ResNet architecture. Hence, we trained BuresGAN on the CIFAR-10 and STL-10 data sets by using the ResNet architecture taken from [69]. In this section, the STL-10 images are rescaled to a resolution of $48 \times 48 \times 3$ according the procedure described in [132, 101, 221], so that the comparison of IS and FID

scores with other works is meaningful. Note that BuresGAN has no parameters to tune, except for the hyperparameters of the optimizers.

The results are displayed in Table 2.5, where the scores of state-of-the-art unconditional GAN models with a ResNet architecture are also reported. In contrast with Section 2.1.3, we report here the best performance achieved at any time during the training, averaged over several runs. To the best of our knowledge, our method achieves a new state of the art inception score on STL-10 and is within a standard deviation of state of the art on CIFAR-10 using a ResNet architecture. The FID score achieved by BuresGAN is nonetheless smaller than the reported FID scores for GANs using a ResNet architecture. A visual inspection of the generated images in Figure 2.3 shows that the high inception score is warranted, the samples are clear, diverse and often recognizable. BuresGAN also performs well on the full-sized STL-10 data set where an inception score of 11.11 ± 0.19 and an FID of 50.9 ± 0.13 is achieved (average and std over 3 runs).

2.1.6 Conclusion

In this work, we discussed an additional term based on the Bures distance which promotes a matching of the distribution of the generated and real data in a feature space $\mathbb{R}^{d_\phi}$. The Bures distance admits both a feature space and kernel based expression, which makes the proposed model time and data efficient when compared to state of the art models. Our experiments show that the proposed methods are capable of reducing mode collapse and, on the real data sets, achieve a clear improvement of sample quality without parameter tuning and without the need for regularization such as a gradient penalty. Moreover, the proposed GAN shows a stable performance over different architectures, data sets and hyperparameters.

2.2 Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks

Abstract

Commonly, machine learning models minimize an empirical expectation. As a result, the trained models typically perform well for the majority of the data but the performance may deteriorate in less dense regions of the dataset. This

issue also arises in generative modeling. A generative model may overlook underrepresented modes that are less frequent in the empirical data distribution. This problem is known as complete mode coverage. We propose a sampling procedure based on ridge leverage scores which significantly improves mode coverage when compared to standard methods and can easily be combined with any GAN. Ridge leverage scores are computed by using an explicit feature map, associated with the next-to-last layer of a GAN discriminator or of a pre-trained network, or by using an implicit feature map corresponding to a Gaussian kernel. Multiple evaluations against recent approaches of complete mode coverage show a clear improvement when using the proposed sampling strategy.

2.2.1 Introduction

Complete mode coverage is a problem of generative models which has been clearly defined and studied in [243]. In layman's terms, a mode is defined as a local maximum of the data probability density. A closely related problem is mode collapse in GANs [64], which happens when a generative model is only capable of generating samples from a subset of all the modes. Multiple GAN variants have been proposed as a solution to this problem, however proposed solutions often assume that every mode has an (almost) equal probability of being sampled, which is often not the case in realistic datasets. Regularly, in critical applications, datasets contain a mixture of different subpopulations where the frequency of each subpopulation can be vastly different. The role of less abundant subpopulations in machine learning data has been discussed recently in [58]. Also, it is often common to presume that an algorithm does not know the abundance of subpopulations. It is however important that a machine learning model performs well on all subpopulations. A standard example is medical data where some rare diseases are less abundant than common diseases. To illustrate the approach presented in this paper, a motivating example containing one majority mode and two minority modes is given in Figure 2.4. When sampling a mini-batch from the Probability Density Function (PDF) p, the side modes can be missed. We observe empirically that this is resolved by sampling from the ridge leverage score (RLS) distribution (see Section 2.2.2), which has been extensively used in randomized linear algebra and kernel methods. Figure 2.4 shows that the samples from the minority modes have larger RLSs. Thus, when sampling from the RLS distribution, there is a higher probability of including the minority modes. This paper is motivated by two situations where minority modes can occur: 1) the observed empirical distribution is different from the true distribution (biased data), and the data needs to be rebalanced. 2) The observed empirical distribution approximates the true distribution sufficiently well, but minority modes consist out of infrequent but very important points,

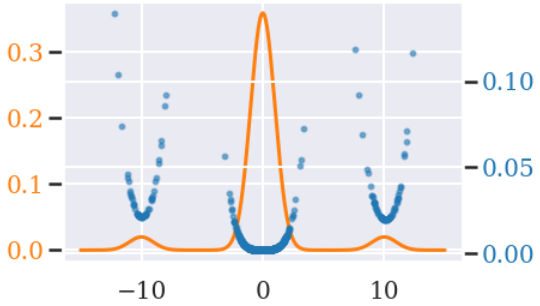


Figure 2.4: Probability Density Function (orange) and RLS of a sample of this PDF (blue). We take the motivating example from [243], which consists of a 1D target PDF p with 1 majority mode and 2 minority modes: $p = 0.9 \cdot \mathcal{N}(0,1) + 0.05 \cdot \mathcal{N}(10,1) + 0.05 \cdot \mathcal{N}(-10,1)$ given in orange. The RLS distribution is calculated using a Gaussian kernel with $\sigma = 3$ and $\gamma = 10^{-3}$. When sampling a mini-batch from the PDF p, the side modes can be missed. This is resolved by sampling from the RLS distribution.

e.g. rare diseases in a medical dataset.

Contribution. When training classical GANs, an empirical expectation of a loss $\mathbb{E}_{x\sim p_d}[\mathcal{L}(x)]$ is optimized in the context of a min-max problem. In this work, we propose a sampling procedure that promotes sampling out of minority modes by using ridge leverage scores. The common algorithmic procedure simulates the empirical distribution over the dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$ by uniformly sampling over this set. We intentionally *bias* or *distort* this process by sampling x_i with probability $p(x_i) \propto \ell_i$, where ℓ_i is the i-th ridge leverage score, defined in Section 2.2.2. Empirical evidence shows that our procedure rebalances the training distribution, as a result, the GAN model generates samples more uniformly over all modes. RLS sampling can easily be applied to any GAN. In particular, using our procedure in combination with a state-of-the-art method for complete mode coverage [243] shows a clear improvement. Finally, RLS sampling is combined with BuresGAN and a state-of-the-art StyleGAN2 with differentiable data augmentations [241], which in both cases improves mode coverage.

Related work. Several works discuss alternative sampling strategies in machine learning. In the context of risk-averse learning, the authors of [36] discuss an adaptive sampling algorithm that performs a stochastic optimization of the Conditional Value-at-Risk (CVaR) of a loss distribution. This strategy promotes models which do not only perform well on average but also on rare data points.

In the context of generative models, AdaGAN [208] is a boosting approach to solve the missing mode problem, where at every step a new component is added into a mixture model by running the GAN training algorithm on a re-weighted sample. A supervised weighting strategy for GANs is proposed in [50]. In this paper, we compare against two state-of-the-art GANs that combat mode collapse, PacGAN [108] and BuresGAN [42]. PacGAN uses a procedure called packing. This modifies the discriminator to make decisions based on multiple samples from the same class, either real or artificially generated. In BuresGAN, an additional diversity metric in the form of the Bures distance between real and fake covariance matrices is added to the generator loss. Note that these methods tackle the traditional mode collapse problem, i.e., the data does not include minority modes. In [238, 239], it was shown that the convergence speed of stochastic gradient descent can be improved by actively selecting mini-batches using DPPs. In [193], coreset-selection is used to create mini-batches with a 'coverage' similar to that of the large batch – in particular, the small batch tries to 'cover' all the same modes as are covered in the large batch.

Before proceeding further, we discuss two main competitors more in-depth. The authors of [50] propose a solution to reduce selection bias in training data named Importance Weighted Generative Networks. A rescaling of the empirical data distribution is performed during training by employing a weighted Maximum Mean Discrepancy (MMD) loss such that the regions where the observed and the target distributions differ are penalized more. Each sample $i \in \{1, \dots, n\}$ is scaled by $1/M(x_i)$, where M is the known or estimated Radon-Nykodym derivative between the target and observed distribution. A version of the vanilla GAN with importance weighting is introduced (IwGAN), as well as the weighting combined with MMDGAN (IwMmdGAN). Another approach to complete mode coverage by [243] and dubbed MwuGAN in this paper, iteratively trains a mixture of generators. At each iteration, the sampling probability is pointwise normalized so that the probability to sample a missing mode is increased. Hence, this generates a sequence of generative models which constitutes the mixture. More precisely, a weight $w_i > 0$ is given for each $i \in \{1, \dots, n\}$ and initialized such that $w_i = p(x_i)$ for some distribution[3] p. Next, a generative model is trained and the probability density $p_g(x_i)$ of each $i \in \{1, \dots, n\}$ is computed. If $p_g(x_i) < \delta p(x_i)$ for some threshold value $\delta \in (0, 1)$, the weight is updated as follows: $w_i \leftarrow 2w_i$, otherwise the weight is not updated. The probability is then recalculated as follows: $p(x_i) = w_i / \sum_j w_j$ for each $i \in \{1, \dots, n\}$. Another generative model is then trained by using $p(x_i)$ and the procedure is repeated.

Classical approach. A GAN consists of a discriminator $D : \mathbb{R}^d \to \mathbb{R}$ and a

[3]In [243], this initial distribution is uniform. We discuss in Section 2.2.3 a choice of weights based on RLSs and initialize MwuGAN with the normalized RLSs in (2.8).

generator $G : \mathbb{R}^\ell \to \mathbb{R}^d$ which are typically defined by neural networks, and parametrized by real vectors. The value $D(x)$ gives the probability that x comes from the empirical distribution, while the generator G maps a point z in the latent space $\mathbb{R}^\ell$ to a point in input space $\mathbb{R}^d$. A typical training scheme for a GAN consists in solving, in an alternating way, the following problems:

$$\begin{aligned} &\max_D \mathbb{E}_{x\sim p_d}[\log(D(x))] + \mathbb{E}_{\tilde{x}\sim p_g}[\log(1 - D(\tilde{x}))], \\ &\min_G -\mathbb{E}_{\tilde{x}\sim p_g}[\log(D(\tilde{x}))], \end{aligned} \tag{2.6}$$

which include the vanilla GAN objective associated with the cross-entropy loss. In (2.6), the first expectation is over the empirical data distribution p_d and the second is over the generated data distribution p_g, implicitly given by the mapping by G of the latent prior distribution $\mathcal{N}(0, \mathbb{I}_\ell)$. The data distribution p_d is estimated using the empirical distribution over the training data $\hat{p}_d(x) = \frac{1}{n}\sum_{x_i\in\mathcal{D}} \delta(x - x_i)$ as follows: $\mathbb{E}_{p_d(x)}[\mathcal{L}(x)] \approx \mathbb{E}_{\hat{p}_d(x)}[\mathcal{L}(x)] = \frac{1}{n}\sum_{x_i\in\mathcal{D}} \mathcal{L}(x_i)$, where $\mathcal{L}$ is a general loss function. As noted by [214], positive weights w_i for $1 \le i \le n$ can be used to construct a weighted empirical distribution $\hat{p}_d^w(x) = \sum_{x_i\in\mathcal{D}} w_i \delta(x - x_i)$, then one can apply a weighting strategy to use samples distributed according to $\hat{p}(x)$ to estimate quantities with respect to $\hat{p}_d^w(x)$ as follows:

$$\mathbb{E}_{\hat{p}_d^w(x)}[\mathcal{L}(x)] = \mathbb{E}_{\hat{p}_d(x)}\left[\frac{\hat{p}_d^w(x)}{\hat{p}_d(x)}\mathcal{L}(x)\right] = \sum_{x_i\in\mathcal{D}} w_i \mathcal{L}(x_i). \tag{2.7}$$

A stochastic procedure is applied for minimizing the above expectation over $\hat{p}_d^w$. In this paper, mini-batches are sampled according to the distribution $\hat{p}_d^w$ with w_i given by the normalized RLSs (2.8) for $1 \le i \le n$.

2.2.2 Sampling with Ridge Leverage Scores

We propose to use a sampling procedure based on ridge leverage scores (RLSs) [4, 116]. RLSs correspond to the correlation between the singular vectors of a matrix and the canonical basis elements. The higher the score, the more *unique* the point. A sample from a minority mode would thus get a higher RLS. These RLSs are used to bias the sampling, which in turn results in a more uniform sampling over all the modes, regardless of the original weight of the mode in the data distribution. Given a feature map $\varphi(\cdot)$, the corresponding kernel function is $K(x, y) = \varphi(x)^\top \varphi(y)$. Let the regularization parameter be $\gamma > 0$. Then, the γ-RLSs are defined for all $1 \le i \le n$ as:

$$\ell_i(\gamma) = \left(K(K + n\gamma\mathbb{I})^{-1}\right)_{ii} = \varphi(x_i)^\top (C + n\gamma\mathbb{I})^{-1}\varphi(x_i), \tag{2.8}$$

where $C = \sum_{i=1}^{n} \varphi(x_i)\varphi(x_i)^\top$ and $K_{ij} = \varphi(x_i)^\top \varphi(x_j)$ for $1 \leq i, j \leq n$. They have both a primal and a dual expression that can be leveraged when the size of the feature map or batch-size respectively are too large. When both the batch-size and feature map dimensions are large, one can rely on fast and reliable approximation algorithms with guarantees such as RRLS [135] and BLESS [175]. The role of $\gamma > 0$ is to filter the small eigenvalues of K in the spirit of Tikhonov regularization. RLSs induce the probability distribution: $p_i = \ell_i / \sum_{j=1}^{n} \ell_j$, for $1 \leq i \leq n$, which is classically used in randomized methods [4]. Figure 2.5 illustrates the interpretation of RLSs on two artificial datasets used in this paper. The datasets consist of a mixture of Gaussians. In the RING example, the first 4 modes are minority modes (starting on top and going further clockwise). In the GRID example, the first 10 modes are minority modes (starting left). Similar to the first illustration (see Figure 2.4), large RLSs are associated with minority modes. More information on the artificial datasets is given in Section 2.2.3.



Figure 2.5: RLS distribution using a Gaussian feature map with $\sigma = 0.15$ and regularization $\gamma = 10^{-3}$ on the unbalanced RING (left) and GRID (right) data. The darker the shade, the higher the RLS. Dark modes correspond to minority modes.

RLS sampling has a rich history in kernel methods and randomized linear algebra but has not been used in the context of GANs. One of the key contributions of this paper is to illustrate the use of RLSs in this setting. To do so, we propose the use of different feature maps so that RLS sampling can be used both for low dimensional and high dimensional data such as images. In what follows, the feature map construction is first discussed. Next, two approximation schemes are introduced.

Choice of the feature map. Three choices of feature maps are considered in this paper to compute leverage scores:

- Fixed implicit feature map. In low dimensional examples, the feature map can be chosen implicitly such that it corresponds to the Gaussian kernel: $\varphi(x)^\top \varphi(y) = \exp\left(-\|x - y\|^2 / \sigma^2\right)$, the bandwidth σ is a hyperparameter.

- Fixed explicit feature map. For image-based data, more advanced similarity metrics are necessary. Therefore, the next-to-last layer of a pre-trained classifier, e.g. the Inception network, is used to extract meaningful features. Note that the classifier does not need to be trained on the exact training dataset, but simply needs to extract useful features.
- Discriminator-based explicit feature map. The feature map can be obtained from the next-to-last layer of the discriminator. Let $D(x) = \sigma(w^\top \varphi_D(x))$, where $w \in \mathbb{R}^m$ contains the last dense layer's weights and σ is the sigmoid function. This feature map is useful in situations where no prior knowledge is available about the dataset.

For a fixed feature map, the RLSs only need to be calculated once before training. The discriminator-based explicit feature map changes throughout the training. Therefore the RLSs are recalculated at every step. Nonetheless, due to approximation schemes which are discussed hereafter, the computational cost stays low. The full algorithm is given in Supplementary Material.

Approximation schemes

Current day models are high dimensional, e.g., a DCGAN yields a feature space $\mathbb{R}^m$ of high dimension $m = 10^3$. Moreover, the size of datasets is commonly thousands up to millions of images. Therefore, two approximation schemes are proposed to speed up the computation of RLSs when using explicit feature maps:

- For the discriminator-based explicit feature map, we propose a two-stage sampling procedure in combination with a Gaussian sketch to reduce the dimension of high dimensional feature maps.
- For the fixed explicit feature map, the well-known UMAP is used to reduce the dimensionality [122].

Two-stage sampling procedure. For the explicit discriminator-based feature map, φ_D has to be re-calculated at each training step. To speed up the sampling procedure, we propose a sampling procedure in two stages. First, a subset of the data is uniformly sampled, e.g. equal to 20 times the desired batch size. Afterward, the RLSs are calculated only for the uniformly sampled subset, which are then used to sample the final batch used for training. This two-stage sampling procedure is similar to the core-set selection used in smallGAN [193]. A first difference is that core-sets are selected by combining a Gaussian sketch and a greedy selection in [193], while we use a randomized approach. Second, in

this reference, cores-sets are used to reduce the batch size to improve scalability. In contrast, RLS sampling is used here to bias the empirical distribution.

Sketching the discriminator feature map. Gaussian sketching is a commonly used method to reduce data dimension and was also used in [193] and [235] to reduce the dimension of large neural nets. Let S be a sketching matrix of size $m \times k$ such that $S = A/\sqrt{k}$ with A a matrix with i.i.d. zero-mean standard normal entries. Consider the following random projection: let a batch be $\{i_1, \ldots, i_b\} \subset \{1, \ldots, n\}$. A random projection of this batch in feature space is then defined as follows:

$$\varphi(x_{i_\ell}) = S^\top \varphi_D(x_{i_\ell}) \in \mathbb{R}^k, \tag{2.9}$$

for all $\ell \in \{1, \ldots, b\}$. This random projection preserves approximately (squared) pairwise distances in the dataset and is motivated by an isometric embedding result in the spirit of Johnson-Lindenstrauss lemma. Let $0 < \epsilon < 1$ and any integer $b > 0$. Let k be an integer such that $k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \log b$. Then, for any set $\{\mathsf{x}_1, \ldots, \mathsf{x}_b\}$ in $\mathbb{R}^m$ there is a map $f : \mathbb{R}^m \to \mathbb{R}^k$ such that for any $\ell, \ell' \in \{1, \ldots, b\}$ we have

$$(1 - \epsilon)\|\mathsf{x}_\ell - \mathsf{x}_{\ell'}\|_2^2 \leq \|f(\mathsf{x}_\ell) - f(\mathsf{x}_{\ell'})\|_2^2 \leq (1 + \epsilon)\|\mathsf{x}_\ell - \mathsf{x}_{\ell'}\|_2^2.$$

The idea of this work is to consider the set of points given by the batch in the discriminator feature space $\mathsf{x}_\ell = \varphi_D(x_{i_\ell})$ for $1 \leq \ell \leq b$. It is proved in [38] that f exists and can be obtained with high probability with a random projection of the form (2.9). For a more detailed discussion, we refer to [146].

Dimensionality reduction of the fixed explicit feature map by UMAP. The Gaussian sketch is a simple and fast method to reduce the dimension of the feature map. This makes it a perfect candidate to reduce the dimension of the proposed discriminator feature map, which has to be recalculated at every iteration. Unfortunately, this speed comes at a price, namely, the gaussian sketch is deemed too simple to reduce the dimension of highly complex models like the Inception network. Therefore, UMAP is proposed [122]. This non-linear dimensionality reduction technique can extract more meaningful features. UMAP is considerably slower than the Gaussian sketch, therefore the use of UMAP is only advised for a fixed feature map like a pre-trained classifier or the Inception network, where the RLSs are only calculated once before training.

2.2.3 Numerical experiments

The training procedure is evaluated on several synthetic and real datasets, where we artificially introduce minority modes. The GANs are evaluated by analyzing the distribution of the generated samples. Ideally, the models should generate samples from every mode as uniformly as possible. A re-balancing effect should be visible. The proposed methods are compared with vanilla GAN, PacGAN, MwuGAN, BuresGAN, IwGAN, and IwMmdGAN. In particular, MwuGAN outperforms AdAGAN [208] on complete mode coverage problems [243]. BuresGAN [42] promotes a matching between the covariance matrices of real and generated data in a feature space defined thanks to the discriminator. Recall that the discriminator is $D(x) = \sigma(w^\top \varphi_D(x))$, where w is a weight vector and the sigmoid function is denoted by σ. The normalization $\bar{\varphi}_D(x) = \varphi_D(x)/\|\varphi_D(x)\|_2$ is used, after the centering of $\varphi_D(x)$. Then, the covariance matrix is defined as follows: $C(p) = \mathbb{E}_{x\sim p}[\bar{\varphi}_D(x)\bar{\varphi}_D(x)^\top]$. The real data and generated data covariance matrices are denoted by $C_d = C(p_d)$ and $C_g = C(p_g)$, respectively. In BuresGAN, the Bures distance is added to the generator loss: $\min_G -\mathbb{E}_{\tilde{x}\sim p_g}[\log(D(\tilde{x}))] + \lambda\mathcal{B}(C_d, C_g)^2$, with the Bures distance $\mathcal{B}(C_d, C_g)^2 = \mathrm{Tr}(C_d + C_g - 2\,(C_d C_g)^{\frac{1}{2}})$, depends implicitly on $\varphi_D(x)$ (see e.g., [121]). The loss of the discriminator remains the same.

Overview of proposed methods. The RLS sampling procedure can easily be integrated into any GAN architecture or model. In this spirit, we used RLS sampling with a classical vanilla GAN (**RLS GAN**) and BuresGAN (**RLS BuresGAN**). The classical BuresGAN has been shown to outperform competitors in mode collapse problems [42]. We noticed empirically that RLS BuresGAN outperformed RLS GAN on the synthetic data (the comparison is shown in Supplementary Material). Therefore we only continue with RLS BuresGAN in the rest of the experiments. Likewise, RLS sampling is combined with MwuGAN, which is considered state-of-the-art in complete mode coverage. The method, called **RLS MwuGAN**, uses RLSs as initial starting weights to sample as opposed to uniform weights. Besides the initialization, the method remains unchanged. The number of generators in the mixture is always displayed in brackets. Unless specified otherwise, the models are trained for 30k iterations with a batch size of 64, by using the Adam [88] optimizer. Unless specified otherwise, we report the means and standard deviations for 10 runs. The largest mean is depicted in black, a * represents significance using a one-tailed Welch's t-test between the best performing proposed model and best performing competitor at a 0.05 confidence level. Further information about the used architectures, hyperparameters, and timings are given in Supplementary Material.

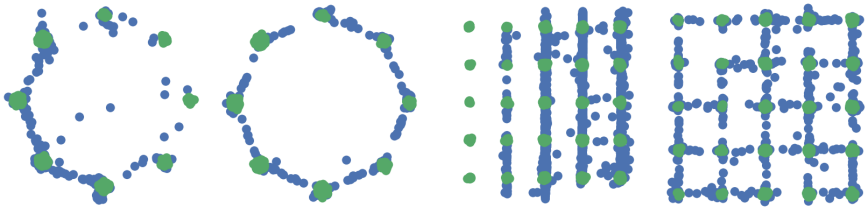


Figure 2.6: Visualization of the generation quality on RING and GRID. Each column shows 2.5k samples from the trained generator in blue and 2.5k samples from the true distribution in green. The vanilla GAN (first and third) does not cover the minority modes. This is not the case for the RLS BuresGAN Discr. (second and fourth).

Synthetic data

Unbalanced versions of two classical synthetic datasets are generated: an unbalanced ring with 4 minority modes (RING) and an unbalanced grid (GRID) with 10 minority modes (see Figure 2.5). RING is a mixture of eight two-dimensional isotropic Gaussians in the plane with means $2.5 \times (\cos((2\pi/8)i), \sin((2\pi/8)i))$ and std 0.05 for $i \in \{1, \dots, 8\}$. The probability of sampling from the first 4 consecutive Gaussians is only 0.05 times the probability of sampling from the last 4 modes. GRID is a mixture of 25 two-dimensional isotropic normals with standard deviation 0.05 and with means on a square grid with spacing 2. The first rectangular blocks of 2×5 adjacent modes are depleted with a factor 0.05.

Evaluation. The evaluation is done by sampling 10k points from the generator network. High-quality samples are within 3 standard deviations of the nearest mode. A mode is covered if there are at least 50 generated samples within 3 standard deviations of the center of the mode. The knowledge of the full Radon-Nikodym derivative M is given to IwMmdGAN and IwGAN by dividing the true probability of each sample (a Gaussian mixture with equal weights) by the adapted Gaussian mixture containing several minority modes. The results of the experiments are given in Table 2.6. For RLS sampling, we use a Gaussian kernel with bandwidth $\sigma = 0.15$, and the discriminator network as feature extractor, both with regularization parameter $\gamma = 10^{-3}$. The models are trained using a fully connected architecture (see Supplementary Material). As the models are rather simple, no dimensionality reduction is needed.

Generated samples from models trained with and without RLS sampling are displayed in Figure 2.6. One can clearly see that training a GAN with uniform sampling results in missing the first 4 minority modes. This is solved by

Table 2.6: Experiments on the synthetic datasets RING and GRID. Two RLS BuresGAN are considered: RLSs calculated with the Gaussian kernel (Gauss.) and the next-to-last layer of the discriminator (Discr.). RLS MwuGAN is initialized with RLSs using an implicit feature map associated with the Gaussian kernel.

	Ring with 8 modes		Grid with 25 modes	
	Nb modes (↑)	% in 3σ (↑)	Nb modes (↑)	% in 3σ (↑)
GAN	5.0(1.1)	**0.92**(0.02)*	8.3(3.4)	0.29(0.3)
PacGAN2	5.4(1.4)	**0.92**(0.03)	10.3(2.6)	0.13(0.02)
BuresGAN	5.8(1.4)	0.76(0.27)	16.7(0.9)	**0.82**(0.01)*
RLS GAN Gauss.	7.4(0.9)	0.86(0.05)	13.8(7.4)	0.51(0.32)
RLS GAN Discr.	7.6(0.8)	0.90(0.02)	20.4(2.6)	0.81(0.03)
IwGAN	**8**(0)	0.85(0.08)	13.4(5.7)	0.29(0.25)
IwMmdGAN	**8**(0)	0.84(0.02)	1.7(5.1)	0.03(0.05)
MwuGAN (15)	7.9(0.3)	0.86(0.02)	15.2(1.7)	0.47(0.11)
RLS MwuGAN (15) (ours)	**8**(0)	0.84(0.06)	22.3(1.9)	0.60(0.1)
RLS BuresGAN Gauss. (ours)	**8**(0)	0.90(0.02)	24.0(1.5)	0.76(0.11)
RLS BuresGAN Discr. (ours)	**8**(0)	0.90(0.02)	**24.4**(0.92)*	0.78(0.06)

using RLS sampling and can be interpreted by comparing the two sampling distributions on Figure 2.7 (uniform) and Figure 2.5 (RLS). The RLSs are larger for samples in minority modes, which results in a more uniform mini-batch over all modes. Note that the RLS sampling procedure, given the feature map, is completely unsupervised and has no knowledge of the desired unbiased distribution. The evaluation metrics in Table 2.6 confirm our suspicions. Only methods designed for complete mode coverage can recover (almost) all modes for the RING dataset. For the unbalanced GRID, only the proposed method has an acceptable performance. Our method even outperforms multiple generator architectures like MwuGAN and RLS MwuGAN, which are considerably more costly to train. Moreover, IwGAN, with full knowledge of M, is not capable of consistently capturing all modes. This was pointed out by the authors in [50]: the method may still experience high variance if it rarely sees data points from a class it wants to boost.

Unbalanced MNIST

For this experiment, we create two unbalanced datasets out of MNIST. The first modified dataset, named UNBALANCED 012-MNIST, consists of only the digits $0, 1$ and 2. The class 2 is depleted so that the probability of sampling

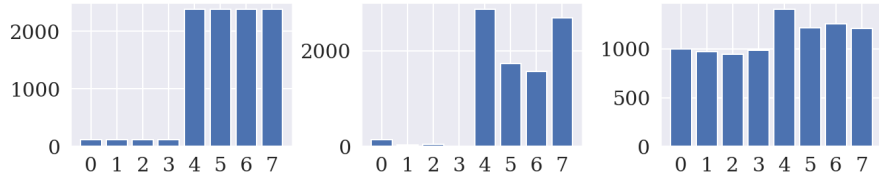


Figure 2.7: RING. Number of training samples in each mode for the RING dataset (left) Generated samples in each mode by a vanilla GAN (middle). Generated samples by RLS BuresGAN Discr. (right). A rebalancing effect is visible.

2 is only 0.05 times the probability of sampling from the digit 0 or 1. The second dataset, named UNBALANCED MNIST, consists of all digits. The classes 0,1,2,3, and 4 are all depleted so that the probability of sampling out of the minority classes is only 0.05 times the probability of sampling from the majority digits. For these experiments, we use a DCGAN architecture. The following metrics are used for performance evaluation: the number of generated digits in each mode, which measures mode coverage, and the KL divergence [125] between the classified labels of the generated samples and a balanced label distribution, which measures sample quality. The mode of each generated image is identified by using a MNIST classifier which is trained up to 98.43% accuracy (see Supplementary Material). The metrics are calculated based on 10k generated images for all the models. For the RLS computation, we use both the discriminator with a Gaussian sketch and the next-to-last layer of the pre-trained classifier with a UMAP dimensionality reduction as a feature map. For the UNBALANCED 012-MNIST, both feature maps are reduced to $k = 25$ and the regularization parameter is $\gamma = 10^{-4}$. In the UNBALANCED MNIST, we take $k = 10$ and $\gamma = 10^{-4}$. An ablation study over different k and γ is given in Supplementary Material. We also compare the performance of the classical MwuGAN, initialized with uniform weights, with RLS MwuGAN where the weights are initialized by the RLSs calculated using the fixed explicit feature map with the same parameters mentioned above. Both methods contain a mixture of 15 GANs, the experiments are repeated 3 times for MwuGAN variants. In our simulations, IwMmdGAN could not be trained successfully with a DCGAN architecture. The Radon-Nikodym derivative M, which is used by IwMmdGAN and IwGAN, is defined as follows: $M_i = 1$ for digits 0 and 1 and $M_i = 0.05$ for digits 2, analogous for the UNBALANCEDMNIST dataset. Only the proposed models trained with RLS sampling are capable of covering all modes consistently. The diversity of images generated by RLS BuresGAN can be visualized in Figure 2.8 where digits from minority modes appear more frequently. A quantitative analysis of mode coverage and sample quality is



Figure 2.8: Generated images from UNBALANCED 012-MNIST by a vanilla GAN (first), by RLS BuresGAN Discr. (second) and generated images from UNBALANCED MNIST by a vanilla GAN (third), by RLS BuresGAN Class. (fourth). The minority digits are generated more frequently in the proposed methods that include RLS sampling.

reported in Tables 2.7 and 2.8. In the UNBALANCED 012-MNIST dataset, there is a clear advantage in using RLS sampling with BuresGAN since mode coverage and the KL divergence are improved compared to the other methods. The second best method is RLS MwuGAN which outperforms RLS with uniform starting weights in KL. For the more difficult UNBALANCED MNIST dataset, using the fixed explicit feature map to calculate the RLSs clearly outperforms other methods.

Table 2.7: Experiments on the UNBALANCED 012-MNIST dataset. Two RLS BuresGAN are considered: RLSs calculated with an explicit feature map obtained from a pre-trained classifier (Class.) and the next-to-last layer of the discriminator (Discr.). RLS MwuGAN is initialized with RLSs using the explicit feature maps obtained from a pre-trained classifier. Minority modes are highlighted in black in the first row.

	Mode 1	Mode 2	**Mode 3**	KL
GAN	4381(172)	5412(179)	129(36)	0.31(0.01)
PacGAN2	4492(237)	5328(242)	123(29)	0.32(0.01)
BuresGAN	4586(287)	5190(292)	142(19)	0.30(0.01)
IwGAN	4368(295)	5414(287)	147(32)	0.32(0.01)
IwMmdGAN	34(12)	0(0)	69(12)	0.56(0.10)
MwuGAN (15)	4886(473)	4865(466)	176(14)	0.31(0.01)
RLS MwuGAN (15) (ours)	3982(218)	4666(164)	870(65)	0.14(0.01)
RLS BuresGAN Class. (ours)	3414(161)	4862(134)	1461(183)	**0.08**(0.01)*
RLS BuresGAN Discr. (ours)	5748(172)	2416(268)	**1566**(293)*	0.16(0.02)

Table 2.8: Experiments on the UNBALANCED MNIST dataset. Two RLS BuresGAN variants are considered: RLSs calculated with an explicit feature map obtained from a pre-trained classifier (Class.) and the next-to-last layer of the discriminator (Discr.). RLS MwuGAN is initialized with RLSs using the explicit feature maps obtained from a pre-trained classifier. Only the number of samples in the minority modes are visualized. The number of samples in the remaining modes are given in Supplementary Material.

	Mode 1	**Mode 2**	**Mode 3**	**Mode 4**	**Mode 5**	KL
GAN	123(22)	137(103)	81(31)	161(97)	161(23)	0.48(0.02)
PacGAN2	109(29)	142(70)	89(23)	147(100)	152(40)	0.48(0.02)
BuresGAN	126(31)	157(97)	108(30)	153(62)	147(26)	0.46(0.02)
IwGAN	117(33)	139(33)	97(31)	212(69)	154(34)	0.46(0.02)
IwMmdGAN	1(1)	0(0)	23(16)	**1140**(367)*	3(3)	1.92(0.1)
MwuGAN (15)	144(29)	113(28)	146(13)	172(18)	167(28)	0.46(0.02)
RLS MwuGAN (15) (ours)	336(47)	283(32)	191(23)	381(38)	276(33)	0.30(0.02)
RLS BuresGAN Class. (ours)	**875**(112)*	**663**(122)*	**360**(198)*	831(59)	**615**(82)*	**0.09**(0.01)*
RLS BuresGAN Discr. (ours)	235(62)	183(141)	264(44)	255(109)	219(54)	0.37(0.02)

Unbalanced CIFAR10

We conclude this section with an experiment on colored images, namely the CIFAR10 dataset. This highly diverse dataset contains 32×32 color images from 10 different classes. We consider two unbalanced variations. The first modified dataset, named UNBALANCED 06-CIFAR10, consists of only the classes 0 and 6 or images of airplanes and frogs respectively. The class 0 is depleted with a factor 0.05. The second dataset, named UNBALANCED 016-CIFAR10, consists of the classes 0,1 and 6. Compared to the previous dataset, we add images from the class automobile. Now, the class 6 consisting of frogs is depleted with a factor 0.05. We show the improvement of RLS sampling in a StyleGAN2 with differentiable data augmentation (StyleGAN2 + Aug.) [241] [4]. By clever use of various types of differentiable augmentations on both real and fake samples, the GAN can match the top performance on CIFAR10 with only 20% training data and is considered state-of-the-art. The StyleGAN2 models are trained for 156k iterations with a mini-batch size of 32 using 'color, translation, and cutout' augmentations, which is suggested by the authors when only part of the CIFAR10 dataset is used. All the other parameters remained the same, only the sampling strategy is changed to RLS sampling in RLS StyleGAN2 + Aug. For the RLS computation, we use the discriminator feature map with Gaussian sketching and a fixed explicit feature map given by the next-to-last layer of the

[4]Code taken from `https://github.com/mit-han-lab/data-efficient-gans`

Inception network where UMAP is used to reduce the dimension. For both the RLSs, the dimension is reduced to $k = 25$ and the regularization parameter is $\gamma = 10^{-4}$. The performance is assessed using 10k generated samples at the end of training by the Inception Score (IS) and the Fréchet inception distance (FID) between the generated fake dataset and the balanced dataset. Mode coverage

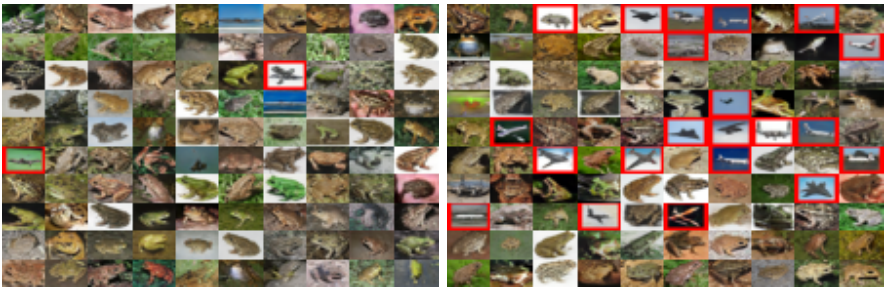


Figure 2.9: Generated images from 06-CIFAR10 by a StyleGAN2 + Aug. (left) and by RLS StyleGAN2 + Aug. (right). Including RLS sampling promotes sampling from the minority class. Generated samples classified as planes are marked by a red border.

is evaluated by the number of generated samples in each class. The class of a generated sample is evaluated by a trained CIFAR10 classifier using a resnet56 type architecture [72] which is trained up to 93.77% accuracy[5]. The results of the experiments are given in Table 2.9, examples of generated images are given in Figure 2.9. Including RLS sampling clearly improves the performance in unbalanced datasets, this is especially the case for the fixed feature map given by the Inception network. The minority mode is oversampled by approximately a factor 10 or even 100 in the case of the unbalanced 06-CIFAR10 and 016-CIFAR10 datasets respectively. Both the IS and FID also improve significantly. Note that the maximum achievable performance for IS and FID is lower when only a subset of classes is included, as pointed out by [178].

2.2.4 Conclusion

We introduced the use of RLS sampling for training GANs. This 'diverse' sampling procedure was motivated by a notion of complete mode coverage in the presence of minority modes. RLS sampling is easy to integrate into any GAN model. Three feature maps have been discussed. An implicit feature map performs well for low-dimensional data. A fixed explicit feature map, such as a

[5]Classifier is available at `https://github.com/gahaalt/ResNets-in-tensorflow2`

Table 2.9: Experiments on the UNBALANCED 06-CIFAR10 and UNBALANCED 016-CIFAR10 dataset. Including RLS sampling in the StyleGAN2 + Aug. clearly improves the performance. Minority modes are highlighted in black in the second row.

	06-CIFAR10				016-CIFAR10				
	Mode 1	Mode 2	IS (↑)	FID (↓)	Mode 1	Mode 2	**Mode 3**	IS (↑)	FID (↓)
StyleGAN2 + Aug.	261	9500	4.8	67.5	4526	5206	18	4.3	48.8
RLS StyleGAN2 + Aug. Disc. (ours)	994	8659	5.7	46.4	4449	5132	139	4.6	44.4
RLS StyleGAN2 + Aug. Class. (ours)	**2438**	7212	**6.2**	**31.3**	4156	4393	**1155**	**5.7**	**27.2**

pre-trained classifier, achieves good results in high-dimensional cases. Lastly, the discriminator can be used as a feature map when no prior knowledge exists about the data. Two approximation methods for the explicit feature maps are also discussed: dimensionality reduction of explicit feature maps and a two-stage sampling procedure to efficiently speed up online RLS computation. We demonstrated empirically that the use of RLS sampling in GANs successfully combats the missing mode problem.

Chapter 3

Restricted Kernel Machines for Time Series

This chapter has been previously published as:

1. A. Pandey, H. De Meulemeester, H. De Plaen, B. De Moor, and J. A. K. Suykens. "Recurrent Restricted Kernel Machines for Time-series Forecasting". In: *Proceedings of ESANN 2022* (2022)

2. A. Pandey, H. De Meulemeester, B. De Moor, and J. A. Suykens. "Multi-view kernel PCA for time series forecasting". In: *Neurocomputing* 554 (2023), p. 126639. ISSN: 0925-2312. DOI: `https://doi.org/10.1016/j.neucom.2023.126639`

Contribution

1. Hannes De Meulemeester helped with developing the methodology, performing the experiments and writing the article in collaboration with the coauthors.

2. Hannes De Meulemeester and Arun Pandey contributed equally. The creation of the model, its implementation, experiments and writing of the article was done collectively.

3.1 Recurrent Restricted Kernel Machines for Time-series Forecasting

Abstract

In this paper, we propose a novel method for time-series modeling and forecasting. It is based on the temporal formulation of Restricted Kernel Machines leading to a dynamical equation in the latent-variables. Forecasting involves finding the next latent variable and then solving a pre-image problem to predict a new-point in the input space. Further, we benchmark our model on several standard data sets against other well-known time-series models.

3.1.1 Introduction

In [198], Suykens proposed a new framework called Restricted Kernel Machines (RKM), which provides a representation of kernel methods with visible and latent variables. This representation has an objective function that is similar to the energy function of Restricted Boltzmann Machines (RBM), thus linking kernel methods with RBMs. Training and prediction requires characterizing the stationary points for the unknowns in the objective. This in turn provides the training and prediction schemes in the kernel methods setting. Restricted Kernel Machines have been previously extended for different tasks such as classification [198], generation [153, 151] and outlier detection [210]. We further extend the RKM framework to time series modeling by introducing a temporal correlation on the latent variables which provides powerful representation learning capabilities, and a novel forecasting method. The formulation draws connections with kernel autoregressive models [84] and Temporal Restricted Boltzmann Machines (TRBM) [197, 145], which are explored in the next sections.

3.1.2 Recurrent Restricted Kernel Machines

Training

Our main objective is to capture the dynamics of a training data set $\mathcal{X}_T$ containing T time steps $\{\boldsymbol{x}_t\}_{t=1}^T \subset \mathscr{X}$. We define a feature map[1] $\phi : \mathscr{X} \to \mathscr{H}$ with $\mathscr{H}$ a (possibly infinite dimensional) Reproducing Kernel Hilbert Space

[1]Throughout our discussion, we assume that the feature vectors are centered in the feature-space *i.e.* $\bar{\phi}(\boldsymbol{x}) = \phi(\boldsymbol{x}) - \mu_\phi$ with $\mu_\phi = \mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{X}}[\phi(\boldsymbol{\xi})]$. Using an implicit formulation, it suffices to notice that $\langle\bar{\phi}(\boldsymbol{x}), \bar{\phi}(\boldsymbol{y})\rangle_{\mathscr{H}} = \langle\phi(\boldsymbol{x}) - \mu_\phi, \phi(\boldsymbol{y}) - \mu_\phi\rangle_{\mathscr{H}} = \langle\phi(\boldsymbol{x}), \phi(\boldsymbol{y})\rangle_{\mathscr{H}} - \langle\mu_\phi, \phi(\boldsymbol{y})\rangle_{\mathscr{H}} -$

(RKHS; see [183] for more details). Such a feature map could be constructed explicitly or implicitly via a kernel function $k(\boldsymbol{x}, \boldsymbol{y}) : \mathscr{X}^2 \to \mathbb{R} : (\boldsymbol{x}, \boldsymbol{y}) \mapsto \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathscr{H}}$. We also define a linear operator[2] $\boldsymbol{V} : \mathbb{R}^s \to \mathscr{H}$ with $s \leq T$ and it's adjoint $\boldsymbol{V}^\star$. Each datapoint $\boldsymbol{x}_t$ will be associated to a latent variable $\boldsymbol{h}_t \in \mathbb{R}^s$ through a pairing term $\langle \phi(\boldsymbol{x}_t), \boldsymbol{V}\boldsymbol{h}_t \rangle_{\mathscr{H}}$. To also capture time dependence, we only add one extra term compared to the original RKM framework [198]: time correlation in the latent space using a set of non-zero lag-dependent coefficients $\mathcal{A}_T = \{a_{t,l} \,|1 \leq t \leq T \text{ and } 0 \leq l < p\}$ with $p \in \mathbb{Z}^+$ a lag parameter (other coefficients are assumed to be 0). Then consider the following objective function with diagonal matrix $\boldsymbol{\Lambda}$:

$$J_T(\boldsymbol{V}, \mathcal{H}_T, \mathcal{X}_T) = \sum_{t=1}^{T} \Big[\overbrace{- \langle \phi(\boldsymbol{x}_t), \boldsymbol{V}\boldsymbol{h}_t \rangle_{\mathscr{H}}}^{\text{feature-space pairing}} \overbrace{- \sum_{l=0}^{p} a_{t,l} \boldsymbol{h}_{t-l}^\top \boldsymbol{h}_t}^{\text{temporal covariance}} + \underbrace{\frac{1}{2}\left(\boldsymbol{h}_t^\top \boldsymbol{\Lambda} \boldsymbol{h}_t + \|\phi(\boldsymbol{x}_t)\|_{\mathscr{H}}^2\right) \Big] + \frac{1}{2}\mathrm{Tr}\left(\boldsymbol{V}^*\boldsymbol{V}\right)}_{\text{regularization}}. \quad (3.1)$$

Interpreting the objective function. The first two terms in (3.1) are similar to the TRBM's energy function [197] which is used (along with bias terms) to define a joint-probability distribution over some visible variables $\{\boldsymbol{x} \in \mathscr{X}\}$ and latent units $\{\boldsymbol{h} \in \{0,1\}^s\}$. It is trained with a maximum-likelihood approach where the gradients are approximated with contrastive divergence. In contrast, we propose to map the data into feature-space and center it to eliminate the need of a bias term. The first term in the objective maximizes the pairing between the visible variables in the feature-space $\{\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathscr{X}\}$ and latent variables $\{\boldsymbol{h} \in \mathbb{R}^s\}$. The second term maximizes the temporal covariance between current and past latent vectors. The regularization terms and constraints are meant to bound the objective.

Solving the objective. Given the visible variables, characterizing the stationary points of $J_T(\boldsymbol{V}, \mathcal{H}_T | \mathcal{X}_T)$ in the latent variables and the pairing linear operator

$\langle \phi(\boldsymbol{x}), \mu_\phi \rangle_{\mathscr{H}} + \langle \mu_\phi, \mu_\phi \rangle_{\mathscr{H}} = k(\boldsymbol{x}, \boldsymbol{y}) - \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{X}}[k(\boldsymbol{\xi}, \boldsymbol{y})] - \mathbb{E}_{\boldsymbol{\zeta} \sim \mathcal{X}}[k(\boldsymbol{x}, \boldsymbol{\zeta})] + \mathbb{E}_{\boldsymbol{\xi}, \boldsymbol{\zeta} \sim \mathcal{X}}[k(\boldsymbol{\xi}, \boldsymbol{\zeta})]$. In practice, we can compute statistics on $\mathcal{X}_T$.

[2]The linear operator $\boldsymbol{V}$ is often referred to as a *matrix* as it only exists explicitly in the case of finite dimensional Hilbert spaces $\mathscr{H}$. It then takes the form $\boldsymbol{V} \in \mathbb{R}^{\dim(\mathscr{H}) \times s}$ and $\boldsymbol{V}^* = \boldsymbol{V}^\top$.

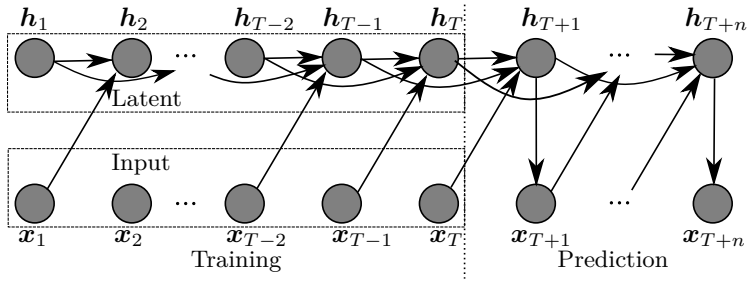


Figure 3.1: Dependency graph of the Recurrent RKM model's *training* (3.3) and *prediction* (3.8) scheme for $a_{t,l} = 1$ if $l = 1$ and $a_{t,l} = 0$ otherwise, and a linear kernel on $\mathscr{X}$.

leads to the following equations for $1 \leq t \leq T$, where $\otimes$ is the outer product:

$$\begin{cases} \dfrac{\partial J_T}{\partial \boldsymbol{V}} = -\sum_{t=1}^{T} \phi(\boldsymbol{x}_t) \otimes \boldsymbol{h}_t + \boldsymbol{V} = 0 \quad \Longrightarrow \quad \boldsymbol{V} = \sum_{t=1}^{T} \phi(\boldsymbol{x}_t) \otimes \boldsymbol{h}_t, & (3.2) \\ \dfrac{\partial J_T}{\partial \boldsymbol{h}_t} = -\boldsymbol{V}^* \phi(\boldsymbol{x}_t) + \boldsymbol{\Lambda} \boldsymbol{h}_t - \left[\sum_{l=0}^{p} a_{t,l} \boldsymbol{h}_{t-l} + \sum_{l=1}^{p} a_{t+l,l} \boldsymbol{h}_{t+l} \right] = 0. & (3.3) \end{cases}$$

Eliminating $\boldsymbol{V}$ from (3.3) using (3.2) gives the following solution

$$\left[\boldsymbol{K}(\mathcal{X}_T) + \boldsymbol{A} \right] \boldsymbol{H}^\top = \boldsymbol{H}^\top \boldsymbol{\Lambda}, \tag{3.4}$$

where $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T] \in \mathbb{R}^{s \times T}$, $\boldsymbol{A}_{i,j} = a_{i,i-j}$ for $i \geq j$ and $\boldsymbol{A}_{i,j} = a_{j-i,j}$ for $i < j$, and kernel matrix $\boldsymbol{K}(\mathcal{X}_T) = [k(\boldsymbol{x}_t, \boldsymbol{x}_{t'})]_{t,t'=1}^{T}$. We can see that any s eigenpairs of $\boldsymbol{K}(\mathcal{X}_T) + \boldsymbol{A}$ satisfies (3.4). The symmetry of $\boldsymbol{A}$ and of the kernel guarantees these eigenvalues to be real. If $\boldsymbol{A}$ is also positive semi-definite, then these eigenvalues are also guaranteed to be positive. An example of such a choice is $a_{t,l} = \exp\left(-l^2/2\sigma_t^2\right)$ for any bandwidth $\sigma_t \in \mathbb{R}^+$. Alternatively, $a_{t,l}$ can be a compactly supported function, for instance, an indicator $a_{t,l} = \mathbf{1}_{\{1,\ldots,p\}}(l)$ (Fig. 3.1 is an example with $p = 1$). Both these choices are however translational invariant, *i.e.* $a_{t,l} = a_l$ for any $a_{t,l} \in \mathcal{A}_T$. In other words, the local effect of time is the same at all time steps.

3.1.3 Prediction

The main idea is to generate $\{\boldsymbol{x}_{T+1}, \ldots, \boldsymbol{x}_{T+n}\}$ for some $n > 0$. To do so, we now work in $\mathcal{X}_{T+n} = \mathcal{X}_T \cup \{\boldsymbol{x}_{T+1}, \ldots, \boldsymbol{x}_{T+n}\}$, $\mathcal{H}_{T+n} = \mathcal{H}_T \cup \{\boldsymbol{h}_{T+1}, \ldots, \boldsymbol{h}_{T+n}\}$

and consider $\mathcal{A}_{T+n}$. This gives the following objective

$$J_{T+n}(\boldsymbol{V}, \mathcal{H}_{T+n}, \mathcal{X}_{T+n}) = \sum_{t=1}^{T+n} \Big[- \langle \phi(\boldsymbol{x}_t), \boldsymbol{V}\boldsymbol{h}_t \rangle_{\mathscr{H}} - \sum_{l=0}^{p} a_{t,l} \boldsymbol{h}_{t-l}^\top \boldsymbol{h}_t + \frac{1}{2} \left(\boldsymbol{h}_t^\top \boldsymbol{\Lambda} \boldsymbol{h}_t + \|\phi(\boldsymbol{x}_t)\|_{\mathscr{H}}^2 \right) \Big] + \frac{1}{2} \mathrm{Tr}\left(\boldsymbol{V}^* \boldsymbol{V}\right). \tag{3.5}$$

Given the learned $\boldsymbol{V}$ from the training, characterizing the stationary points of $J_{T+n}\left(\mathcal{X}_{T+n}, \mathcal{H}_{T+n} \,|\boldsymbol{V}\right)$ in terms of visible and latent variables gives for $1 \leq t \leq T+n$

$$\begin{cases} \dfrac{\partial J_{T+n}}{\partial \phi\left(\boldsymbol{x}_t\right)} = -\boldsymbol{V}\boldsymbol{h}_t + \phi\left(\boldsymbol{x}_t\right) = 0 \quad \Longrightarrow \quad \phi\left(\boldsymbol{x}_t\right) = \boldsymbol{V}\boldsymbol{h}_t, & (3.6) \\ \dfrac{\partial J_{T+n}}{\partial \boldsymbol{h}_t} = -\boldsymbol{V}^* \phi(\boldsymbol{x}_t) + \boldsymbol{\Lambda}\boldsymbol{h}_t - \left[\sum_{k=0}^{p} a_{t,l} \boldsymbol{h}_{t-l} + \sum_{l=1}^{p} a_{t+l,l} \boldsymbol{h}_{t+l} \right] = 0. & (3.7) \end{cases}$$

We first notice that $\phi\left(\boldsymbol{x}_t\right) = \boldsymbol{V}\boldsymbol{h}_t$ is true for all $1 \leq t \leq T$. Furthermore, we also have $\partial J_{T+n}/\partial \boldsymbol{h}_t = \partial J_T/\partial \boldsymbol{h}_t$ for all $1 \leq t \leq T-p$. Using $\partial J_{T+n}/\partial \boldsymbol{h}_t = 0$ (3.7) and the obtained $\boldsymbol{V}$ (3.2), with $t = T-p+1$, we can find an expression for $\boldsymbol{h}_{T+1}$. Iteratively, we can find an expression for $\boldsymbol{h}_{T+m}$ with $t = T-p+m$, until $m = n$:

$$a_{T+m,t} \boldsymbol{h}_{T+m} = \Big[\boldsymbol{H}\boldsymbol{A}\boldsymbol{H}^\top - a_{T+m-p,0} \mathbb{I}_s \Big] \boldsymbol{h}_{T+m-p} - \Big[\sum_{l=1}^{p} a_{T+m-p,l} \boldsymbol{h}_{T+m-p-l} + \sum_{l=1}^{p-1} a_{T+m-p+l,l} \boldsymbol{h}_{T+m-p+l} \Big]. \tag{3.8}$$

This can now be used in (3.6) to find $\phi\left(\boldsymbol{x}_{T+m}\right)$, again with $t = T+m$:

$$\phi(\boldsymbol{x}_{T+m}) = \boldsymbol{V}\boldsymbol{h}_{T+m} = \sum_{t'=1}^{T} \phi(\boldsymbol{x}_{t'}) \boldsymbol{h}_{t'}^\top \boldsymbol{h}_{T+m}. \tag{3.9}$$

Finally, to obtain new data points $\{\boldsymbol{x}_t\}_{t=T+1}^{T+n}$ in the input space, the *pre-image* problem on (3.9) needs to be solved.

Solving the pre-image problem. An advantage of using a kernel function, $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathscr{H}}$, is that all computations can be implicitly performed in feature space and the exact mapping $\phi : \mathscr{X} \to \mathscr{H}$ is not required. Working with an implicit feature map however gives rise to the pre-image problem. Given a point $\boldsymbol{\psi} \in \mathscr{H}$, find $\boldsymbol{x} \in \mathscr{X}$ such that $\boldsymbol{\psi} = \phi(\boldsymbol{x})$. This pre-image

problem is known to be ill-posed as the exact pre-image might not exist [185]. Instead, an optimization problem is considered to find the approximate pre-image $\tilde{\boldsymbol{x}} = \arg\min_{\tilde{\boldsymbol{x}} \in \mathcal{X}} \|\boldsymbol{\psi} - \phi(\tilde{\boldsymbol{x}})\|^2_{\mathcal{H}}$. We employ two different pre-image methods in this work to solve (3.9): kernel smoother [187] and kernel ridge regression [225].

Computational complexity. The eigendecomposition during training (see (3.4)) requires $\mathcal{O}(T^3)$ operations, the complexity of the predictions in latent space $\mathcal{O}(p)$ and in input space with the kernel-smoother $\mathcal{O}(T)$, whereas training the kernel-ridge regression $\mathcal{O}(T^3)$ since it involves solving a linear-system.

3.1.4 Experiments

We illustrate the representation learning capabilities by considering a simple sine wave as input to the RRKM model and exploring its latent space. Fig. 3.2 shows the latent space embedding of the learned sine wave and evolution of forecasted latent variables. Dynamics in the data are well represented in the latent space and the forecasted latent variables continue to follow the training trajectory. In Fig. 3.3, we perform an ablation study on the Santa Fe data set to identify the effect of hyper-parameters on the forecasting performance. We vary bandwidths σ_x, σ_t and latent-space dimension s. The study shows that σ_t captures phase-shift, σ_x captures amplitude and s capture higher and lower frequencies.

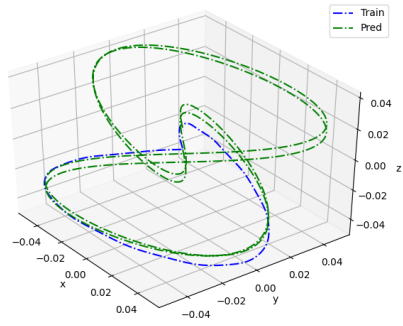


Figure 3.2: Training and predicted latent variables of a sinusoidal data set.

The proposed model is compared to a recurrent neural network (RNN) and an ARMA model which are two of the most popular methods used in time series forecasting. On each data set [3], and method, hyperparameter tuning has been performed and the result of the best set of parameters, quantified as the mean squared error, is shown in Table 3.1. For all methods, the entire validation set is forecasted, in recursive fashion, starting from the end of the training set.

[3] `https://rdrr.io/cran/TSPred/man/SantaFe.A.html`, `https://archive.ics.uci.edu/ml/datasets/Hungarian+Chickenpox+Cases`, `https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction`, `https://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set`.

When comparing to the baseline methods, RRKM is comparable or better. The RNN can have a better result, however, due to its stochastic nature, its performance has high variability while the RRKM is deterministic for the same parameters.

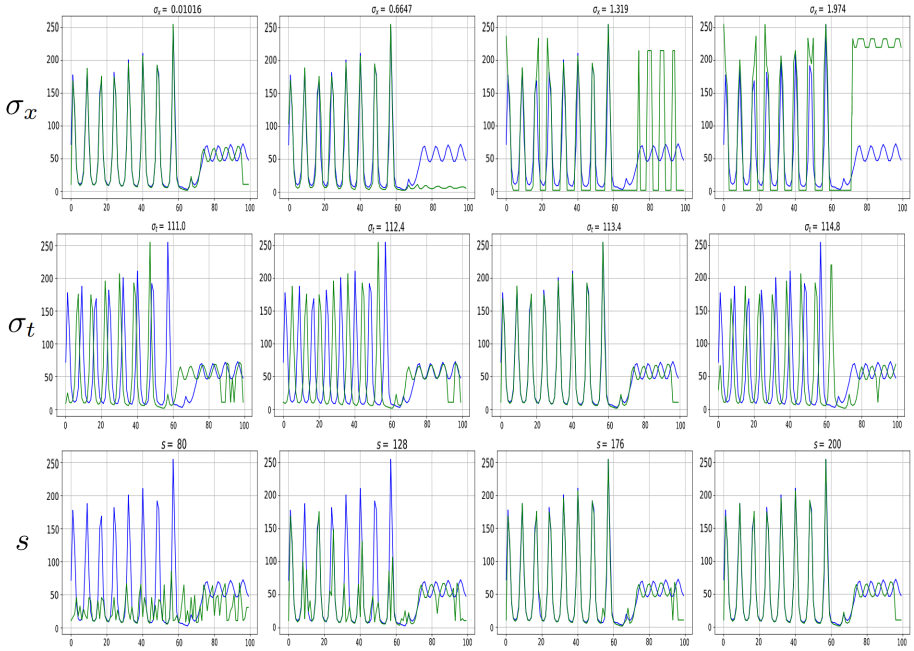


Figure 3.3: Ablation study on the Santa Fe laser data set where the blue curve is the ground-truth and the green curve is the prediction.

Table 3.1: Mean squared error on the forecasted data. Standard deviation for 10 iterations between brackets for the stochastic models.

Data	**RNN**	**ARMA**	**RRKM (Ours)**
Santa Fe	3075.06 (±794.10)	2224.55	**119.06**
Chickenpox	34329.95 (±9513.07)	23571.35	**20716.91**
Energy	16002.11 (±1809.89)	24797.40	**12764.097**
Turbine	2401.12 (±644.53)	1317.67	**1299.915**

3.1.5 Conclusion

In this work, we introduced the recurrent restricted kernel machine model. This framework provides new insights and ideas for time series modeling including latent space dynamics and a novel forecasting method. For future work, we believe a further exploration of the representation learning capabilities in latent space can provide new ways to interpret the data. Additionally, besides the topics mentioned in this work, the framework can be extended towards other tasks involving time series such as denoising, handling missing values and classification.

3.2 Multi-view Kernel PCA for Time series Forecasting

Abstract

In this paper, we propose a kernel principal component analysis model for multi-variate time series forecasting, where the training and prediction schemes are derived from the multi-view formulation of Restricted Kernel Machines. The training problem is simply an eigenvalue decomposition of the summation of two kernel matrices corresponding to the views of the input and output data. When a linear kernel is used for the output view, it is shown that the forecasting equation takes the form of kernel ridge regression. When that kernel is non-linear, a pre-image problem has to be solved to forecast a point in the input space. We evaluate the model on several standard time series datasets, perform ablation studies, benchmark with closely related models and discuss its results.

3.2.1 Introduction

Kernel methods have seen great success in many applications with very-high dimensional data but low number of samples, and are therefore one of the most popular non-parametric models. In critical machine learning applications, kernel methods are preferred due to their strong theoretical foundation in learning theory [219, 185, 199, 164]. Kernel methods map the data into a high-dimensional (possibly infinite) feature space by using the kernel trick. This kernel trick allows for natural, non-linear extensions to the traditional linear methods in terms of a dual representation using a suitable kernel function. This led to numerous popular methods such as kernel principal component analysis [184], kernel fisher discriminant analysis [127] and the least-squares support vector machine [199].

However, in the case of large-scale problems, kernel methods fall behind deep learning techniques due to their time and memory complexity. This also holds in the time series analysis and forecasting domain, which has recently been dominated by specialized deep neural network models [143, 24, 244, 30].

Attempts have been made to combine kernel and deep learning methods [33, 229], especially for specific cases such as with deep gaussian processes [37] and multi-layer support vector machines [227]. Recently, a new unifying framework, named Restricted Kernel Machines (RKM), was proposed [198] that attempts to bridge kernel methods with deep learning. The Lagrangian function of the Least-

Squares Support Vector Machine (LS-SVM) is similar to the energy function of Restricted Boltzmann Machine (RBMs), thereby drawing a link between kernel methods and RBMs; hence the name Restricted Kernel Machines.

Contribution: In this work, we propose a novel kernel autoregressive time series forecasting model based on the RKM framework, where the training problem is the eigenvalue decomposition of two kernel matrices. Additionally, we use the same objective function to derive a prediction scheme to recursively forecast several steps ahead in the future. Further, we experimentally compare the model with several closely related models on multiple publically available time series datasets.

3.2.2 Related Works

Energy-based models: Energy-based models encode dependencies between visible and latent variables using the so-called energy function. Such a function is used to model joint-probability distribution over visible and latent variables of the system. Such a model is trained by finding a configuration with the lowest total energy. Energy based models have the advantage over probabilistic methods that they do not require proper normalization, which in turn allows for more freedom when designing a model. One of the most well known energy-based models is the Restricted Boltzmann Machine [180]. Among other applications, this model has been extended for time series modeling [196, 197]. For an excellent review of Energy-based Models and Restricted Boltzmann Machines for time series modeling, see [144, 145].

Kernel methods for time series: Kernel methods have long been a staple in the time series analysis toolkit. Thanks to Mercer's theorem [123], these methods can implicitly work in very high dimensional spaces and can often allow for complicated non-linear problems to be turned into convex optimization problems.

In the case of time series forecasting, one can change the traditional kernel ridge regression formulation into an online variant where only a number of data points are available at each time step. The solution to this problem can then be iteratively updated for each new time step. This technique is known as Kernel Recursive Least-Squares [55] (KRLS). One disadvantage to this method is that the model increases in size for each new data point that gets introduced. Various solutions for this have been proposed that attempt to limit the final model size [216, 218]. Alternatively, a Bayesian formulation for KRLS exists which also includes a forgetting mechanism [217].

Support vector machines [134, 199] are a particular popular type of kernel

machine, since training an SVM corresponds with solving a convex optimization problem with sparse solutions. The standard, static regression variants of support vector machines can be used in feedforward fashion for time series forecasting. Alternatively, explicitly recursive extensions have been formulated [200]. The disadvantage is that the resulting optimization problems become non-convex.

Gaussian processes [228] are the extension of kernel methods to the probabilistic setting. Unlike support vector machines, Gaussian processes are not sparse, which can be problematic for large scale problems. Various solutions have been proposed to reduce the effective number of data points and train on large datasets, such as by using inducing variables [207] or by performing the computations on subsets of the data [47]. [95] showed that a Gaussian process could be formulated as a latent variable model. This allowed for hierarchical, deep, Gaussian processes [37].

Restricted Kernel Machines

Restricted Kernel Machines (RKM) [198] is a general kernel methods framework characterized by visible and hidden units. Using the Fenchel-Young conjugate for quadratic functions, RKMs make the connection between Least-Squares Support Vector Machines [199] and Restricted Boltzmann Machines [180]. The workhorse of restricted kernel machines is kernel PCA, of which the energy function can be defined in the RKM framework using visible ($\boldsymbol{v}_i \in \mathbb{R}^d$), latent ($\boldsymbol{h}_i \in \mathbb{R}^s$) units paired by the interconnection matrix $\boldsymbol{W} \in \mathbb{R}^{d \times s}$ as follows:

$$J(\boldsymbol{W}, \boldsymbol{h}_i) = \sum_{i=1}^{n} \left[-\langle \boldsymbol{v}_i, \boldsymbol{W}\boldsymbol{h}_i \rangle_{\mathscr{H}} + \frac{1}{2} \boldsymbol{h}_i^\top \boldsymbol{\Lambda} \boldsymbol{h}_i \right] + \frac{1}{2} \operatorname{Tr}(\boldsymbol{W}^\top \boldsymbol{W}). \tag{3.10}$$

Here $\boldsymbol{\Lambda}$ is the diagonal matrix capturing the strength of the latent components, and $\mathscr{H}$ denotes the Reproducing Kernel Hilbert Space. The stationary points of the energy function are given by:

$$\begin{cases} \dfrac{\partial J}{\partial \boldsymbol{h_i}} = 0 \implies \boldsymbol{W}^\top \boldsymbol{v}_i = \boldsymbol{\Lambda} \boldsymbol{h_i}, \quad \forall i \\ \dfrac{\partial J}{\partial \boldsymbol{W}} = 0 \implies \boldsymbol{W} = \displaystyle\sum_{i=1}^{n} \boldsymbol{v}_i \boldsymbol{h}_i^\top. \end{cases}$$

Eliminating $\boldsymbol{W}$ from above gives the following eigenvalue problem:

$$\boldsymbol{K}\boldsymbol{H}^\top = \boldsymbol{H}^\top \boldsymbol{\Lambda} \tag{3.11}$$

with $\boldsymbol{H} = \begin{bmatrix}\boldsymbol{h}_1 \dots \boldsymbol{h}_n\end{bmatrix} \in \mathbb{R}^{s\times n}$, $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_s) \in \mathbb{R}^{s\times s}$ where generally $s \leq n$, indicating the number of components and $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle_{\mathscr{H}}$ are the elements of the kernel matrix $\boldsymbol{K}$. The above closed-form solution is the dual form, and hence the matrix $\boldsymbol{W}$ need not be explicitly computed. Training an RKM model consists of solving an eigenvalue problem. This is in contrast with Restricted Boltzmann Machines where the model is trained by approximating the gradient of the log-likelihood of $\boldsymbol{W}$ by using methods such as contrastive divergence [77].

The objective function $J(\boldsymbol{W}, \boldsymbol{h}_i)$ in Eq. (3.10) forms the basis for more specific RKM models which can be used in downstream tasks. Additionally, a key advantage of characterizing kernel methods by visible and hidden units is that deeper, multi-layer, models can be developed similarly to deep Boltzmann machines [179].

Classification and regression can be performed by introducing the primal formulation of LS-SVM into the objective function [198]. This is motivated by the fact that kernel PCA corresponds to a one-class LS-SVM problem with zero target value around which the variance is maximized [201]. Classification using RKMs has been extended to multi-view classification using a tensor-based representation of the methods [81]. RKMs have been shown to be powerful feature extractors and can be used for disentangled representation learning [210, 230]. This, in turn, allows them to be used in outlier and out-of-distribution detection tasks [209]. Finally, the latent space of an RKM can be sampled from to generate new samples from the learned distribution [187, 152, 153, 150].

One aspect that was missing, until now, was a general time series model situated in the RKM setting. This work attempts to fill this gap by introducing a framework with autoregressive RKMs. This idea was briefly introduced in [148] where an RKM model with autoregression in the latent space was discussed. In the next section, we generalize this method and introduce a complete framework for time series analysis using restricted kernel machines.

3.2.3 Proposed Model

Our objective is to capture the dynamics of the multi-variate time series $\{\boldsymbol{x}_i\}_{i=1}^n, \boldsymbol{x}_i \in \mathbb{R}^d$ containing n time steps. Given the lag $p \in \mathbb{Z}^+$, we transform the given time series such that the input vector is a vector of lagged values and the corresponding output vector is one-step ahead data-point. Let the input data-vectors be $\{[\boldsymbol{x}_i^{(1)}, \dots, \boldsymbol{x}_{i-p}^{(1)}, \boldsymbol{x}_i^{(d)}, \dots, \boldsymbol{x}_{i-p}^{(d)}]\}_{i=p+1}^{n-1} \subset \mathcal{X}$ and the corresponding target values be $\{\boldsymbol{x}_{i+1}\}_{i=p+1}^{n-1} \subset \mathcal{Y}$. This model structure is employed to model the dynamics of the system which enables us to use static regression models for

training. However, for prediction, we employ the *recursive* model such that the predictions are used as input for further predictions $\hat{\boldsymbol{y}} = f(\hat{\boldsymbol{y}}_{i-1}, \ldots, \hat{\boldsymbol{y}}_{i-p})$.

We define feature maps $\phi : \mathscr{X} \to \mathscr{H}_x$, $\psi : \mathscr{Y} \to \mathscr{H}_y$ with $\mathscr{H}_x, \mathscr{H}_y$ indicating the Reproducing Kernel Hilbert Space[4] (RKHS; see [183] for more details). Such a feature map could be constructed explicitly or implicitly via a kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j) : \mathscr{X}^2 \to \mathbb{R} : (\boldsymbol{x}_i, \boldsymbol{x}_j) \mapsto \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathscr{H}_x}$. Further we define two linear operators[5] $\boldsymbol{U}_x : \mathbb{R}^s \to \mathscr{H}_x$ and $\boldsymbol{U}_y : \mathbb{R}^s \to \mathscr{H}_y$, $s \leq n$ with their ad-joints $\boldsymbol{U}_x^\top : \dim(\mathscr{H}_x) \to \mathbb{R}^s$, $\boldsymbol{U}_y^\top : \dim(\mathscr{H}_y) \to \mathbb{R}^s$. Data points $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ will be associated to a latent variable $\boldsymbol{h}_t \in \mathbb{R}^s$ through the pairing terms $\langle \phi(\boldsymbol{x}_i), \boldsymbol{U}_x \boldsymbol{h}_i \rangle_{\mathscr{H}_x}$ and $\langle \psi(\boldsymbol{y}_i), \boldsymbol{U}_y \boldsymbol{h}_i \rangle_{\mathscr{H}_y}$.

Training

The training objective is defined as follows:

$$J_n(\boldsymbol{U}_x, \boldsymbol{U}_y, \mathcal{H}_n) = \sum_{i=1}^{n} \Big[\overbrace{- \langle \phi(\boldsymbol{x}_i), \boldsymbol{U}_x \boldsymbol{h}_i \rangle_{\mathscr{H}_x}}^{1^{\text{st}} \text{ view}} \overbrace{- \langle \psi(\boldsymbol{y}_i), \boldsymbol{U}_y \boldsymbol{h}_i \rangle_{\mathscr{H}_y}}^{2^{\text{nd}} \text{ view}} + \frac{1}{2} \big[\boldsymbol{h}_i^\top \boldsymbol{\Lambda} \boldsymbol{h}_i + \underbrace{\|\phi(\boldsymbol{x}_i)\|_{\mathscr{H}_x}^2 + \|\psi(\boldsymbol{y}_i)\|_{\mathscr{H}_y}^2 \big] \Big] + \frac{1}{2} \big[\|\boldsymbol{U}_x\|_{\text{F}}^2 + \|\boldsymbol{U}_y\|_{\text{F}}^2 \big]}_{\text{regularization}} \quad (3.12)$$

The first two terms in the objective maximizes the pairing between the visible variables in the feature-space $\{\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathscr{X}\}$, $\{\psi(\boldsymbol{y}) : \boldsymbol{y} \in \mathscr{Y}\}$, latent variables $\{\boldsymbol{h} \in \mathbb{R}^s\}$ and $\boldsymbol{\Lambda}$ is the diagonal matrix capturing the strength of the latent components. In this way, we map the points from the two data-sources to a single point in the latent space. The objective function itself is non-convex for certain values of $\boldsymbol{\Lambda}$ [3]. The solutions are found from analyzing the saddle-points. This yields a closed form solution in the form of a KPCA problem. Substituting back the solutions, we obtain J_n as 0.

Solving the objective Given the visible variables, characterizing the stationary points of $J(\boldsymbol{U}_x, \boldsymbol{U}_y, \mathcal{H}_T | \mathcal{X}_T)$ in the pairing linear operators and the latent

[4] Throughout our discussion, we assume that the feature vectors are centered in the feature-space *i.e.* (possibly infinite dimensional) $\bar{\phi}(\boldsymbol{x}) = \phi(\boldsymbol{x}) - \mu_\phi$ with $\mu_\phi = \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{X}} [\phi(\boldsymbol{\xi})]$. Using an implicit formulation, it suffices to notice that $\langle \bar{\phi}(\boldsymbol{x}), \bar{\phi}(\boldsymbol{y}) \rangle_{\mathscr{H}} = \langle \phi(\boldsymbol{x}) - \mu_\phi, \phi(\boldsymbol{y}) - \mu_\phi \rangle_{\mathscr{H}} = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathscr{H}} - \langle \mu_\phi, \phi(\boldsymbol{y}) \rangle_{\mathscr{H}} - \langle \phi(\boldsymbol{x}), \mu_\phi \rangle_{\mathscr{H}} + \langle \mu_\phi, \mu_\phi \rangle_{\mathscr{H}} = k(\boldsymbol{x}, \boldsymbol{y}) - \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{X}} [k(\boldsymbol{\xi}, \boldsymbol{y})] - \mathbb{E}_{\boldsymbol{\zeta} \sim \mathcal{X}} [k(\boldsymbol{x}, \boldsymbol{\zeta})] + \mathbb{E}_{\boldsymbol{\xi}, \boldsymbol{\zeta} \sim \mathcal{X}} [k(\boldsymbol{\xi}, \boldsymbol{\zeta})]$. In practice, we can compute statistics on $\mathcal{X}$ (see C.1).

[5] The linear operator $\boldsymbol{U}_x$ is often referred to as a *matrix* as it only exists explicitly in the case of finite dimensional Hilbert spaces $\mathscr{H}$. It then takes the form $\boldsymbol{U}_x \in \mathbb{R}^{\dim(\mathscr{H}_x) \times s}$.

variables leads to the following equations for $1 \leq i \leq n$:

$$\begin{cases} \dfrac{\partial J}{\partial \boldsymbol{h}_i} = 0 \Rightarrow \boldsymbol{\Lambda} \boldsymbol{h}_i = \boldsymbol{U}_x^\top \phi(\boldsymbol{x}_i) + \boldsymbol{U}_y^\top \psi(\boldsymbol{y}_i) & (3.13) \\ \dfrac{\partial J}{\partial \boldsymbol{U}_x} = 0 \Rightarrow \boldsymbol{U}_x = \displaystyle\sum_{i=1}^{n} \phi(\boldsymbol{x}_i) \boldsymbol{h}_i^\top & (3.14) \\ \dfrac{\partial J}{\partial \boldsymbol{U}_y} = 0 \Rightarrow \boldsymbol{U}_y = \displaystyle\sum_{i=1}^{n} \psi(\boldsymbol{y}_i) \boldsymbol{h}_i^\top . & (3.15) \end{cases}$$

Eliminating $\boldsymbol{U}_x, \boldsymbol{U}_y$ from the above equations and denoting $\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}_1 \dots \boldsymbol{h}_n \end{bmatrix} \in \mathbb{R}^{s \times n}$ and $\boldsymbol{\Lambda} = \text{diag}(\lambda_i) \in \mathbb{R}^{s \times s}$, we get the following eigenvalue problem:

$$\Big[\boldsymbol{K}(\mathcal{X}) + \boldsymbol{K}(\mathcal{Y}) \Big] \boldsymbol{H}^\top = \boldsymbol{H}^\top \boldsymbol{\Lambda}, \tag{3.16}$$

where the elements of $\boldsymbol{K}(\mathcal{X})_{i,j} = k_x(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{K}(\mathcal{Y})_{i,j} = k_y(\boldsymbol{y}_i, \boldsymbol{y}_j)$. Further, $\boldsymbol{K}(\mathcal{X})$ and $\boldsymbol{K}(\mathcal{Y})$ are Hermitian matrices and their sum is also Hermitian. *Spectral theorem* guarantees the existence of real eigenvalues when diagonalized by a unitary matrix. Moreover these eigenvalues are non-negative, since these matrices are positive semi-definite.

Remarks The proposed model is inspired by [148], however there are three important differences. (i) The data in the proposed model is in the auto-regressive format. This enables us to formulate the problem in the well-studied Non-linear Auto-Regressive (NAR) model setting. (ii) The second-view $\boldsymbol{K}(\mathcal{Y})$ is now a kernel matrix i.e. a positive semi-definite matrix. This was not a constraint with the *time-kernel* in [148]. Moreover, $\boldsymbol{K}(\mathcal{Y})$ implicitly captures the weights on the past latent-variables via a kernel function instead of hand-crafting it. (iii) The prediction scheme in [148] used past latent variables to predict the next latent variable. This was sufficient to simulate the dynamics forward after the training-set, however there was no way to start with *any* given initial state vector. In contrast, here we require the initial state vector from the input space to predict the dynamics forward.

Finally, the proposed model is defined with 2 views, one on the input and one on the output. However, it's straightforward to extend it towards multiple views by adding additional pairing terms with corresponding regularization to the objective 3.12. This results in an additional kernel matrix term in (3.16). For example, one could imagine multiple views on the input, where for each view the lag is varied.

Prediction

The main goal is to forecast ahead of the training set i.e. find $\{\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_{n'}\}$, $n' > n$. To do so, we now work in $\mathcal{X}_{n'} = \mathcal{X}_n \cup \{\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_{n+n'}\}$, $\mathcal{H}_{n'} = \mathcal{H}_n \cup \{\boldsymbol{h}_{n+1}, \ldots, \boldsymbol{h}_{n+n'}\}$ and consider $\mathcal{A}_{n'}$. This gives the following objective

$$J_{n'}(\boldsymbol{U}_x, \boldsymbol{U}_y, \mathcal{H}_{n'}) = \sum_{i=1}^{n'} \Big[- \langle \phi(\boldsymbol{x}_i), \boldsymbol{U}_x \boldsymbol{h}_i \rangle_{\mathscr{H}_x} - \langle \psi(\boldsymbol{y}_i), \boldsymbol{U}_y \boldsymbol{h}_i \rangle_{\mathscr{H}_y} + \frac{1}{2} \big[\boldsymbol{h}_i^\top \boldsymbol{\Lambda} \boldsymbol{h}_i + \|\phi(\boldsymbol{x}_i)\|^2_{\mathscr{H}_x} + \|\phi(\boldsymbol{y}_i)\|^2_{\mathscr{H}_y} \big] \Big] + \frac{1}{2} \Big[\|\boldsymbol{U}_x\|^2_{\mathrm{F}} + \|\boldsymbol{U}_y\|^2_{\mathrm{F}} \Big] \quad (3.17)$$

Given the learned $\boldsymbol{U}_x, \boldsymbol{U}_y$ from the training, characterizing the stationary points of $J_{n'}$ $(\mathcal{X}_{n'}, \mathcal{H}_\prime \,|\boldsymbol{U}_x, \boldsymbol{U}_y)$ in terms of visible and latent variables gives for $1 \leq i \leq n'$:

$$\begin{cases} \dfrac{\partial J}{\partial \phi(\boldsymbol{x}_i)} = 0 \Rightarrow \phi(\boldsymbol{x}_i) = \boldsymbol{U}_x \boldsymbol{h}_i & (3.18) \\ \dfrac{\partial J}{\partial \psi(\boldsymbol{y}_i)} = 0 \Rightarrow \psi(\boldsymbol{y}_i) = \boldsymbol{U}_y \boldsymbol{h}_i & (3.19) \\ \dfrac{\partial J}{\partial \boldsymbol{h}_i} = 0 \Rightarrow \boldsymbol{\Lambda} \boldsymbol{h}_i - \boldsymbol{U}_y^\top \psi(\boldsymbol{y}_i) = \boldsymbol{U}_x^\top \phi(\boldsymbol{x}_i). & (3.20) \end{cases}$$

Using the above equations, we can obtain an expression for $\boldsymbol{h}_i$ with $\phi(\boldsymbol{x}_i)$:

$$\boldsymbol{h}_i = \big(\boldsymbol{\Lambda} - \boldsymbol{U}_y^\top \boldsymbol{U}_y \big)^{-1} \boldsymbol{U}_x^\top \phi(\boldsymbol{x}_i). \quad (3.21)$$

Substituting it back in Eq. (3.19), we obtain an expression for output in the feature-space:

$$\psi(\boldsymbol{y}_i) = \boldsymbol{U}_y \big(\boldsymbol{\Lambda} - \boldsymbol{U}_y^\top \boldsymbol{U}_y \big)^{-1} \boldsymbol{U}_x^\top \phi(\boldsymbol{x}_i). \quad (3.22)$$

Depending on whether $\psi(\cdot)$ is linear or non-linear, the solution depends on two cases, which are discussed below.

Case I: Non-linear k_x, linear k_y: In this case, Eq. (3.22) reduces to the following variant of kernel ridge regression $\boldsymbol{y}_i = \big(\sum_{j=1}^n \tilde{\boldsymbol{h}}_j k_x(\boldsymbol{x}_j, \boldsymbol{x}_i) \big)$, where $\tilde{\boldsymbol{h}} = \boldsymbol{U}_y \big(\boldsymbol{\Lambda} - \boldsymbol{U}_y^\top \boldsymbol{U}_y \big)^{-1} \boldsymbol{h}_j$. The way in which those coefficients are obtained, differs between our method and kernel ridge regression.

Case II: Non-linear k_x, non-linear k_y: If the feature-map $\psi(\cdot)$ is explicitly known and *invertible*, then after the inversion, we have the desired output $\boldsymbol{y}_i$. However typically in kernel methods, for instance when using Gaussian kernel, $\psi(\cdot)$ is unknown. As a consequence, the closed-form expression in Eq. (3.22)

cannot be computed. In such situations, we leverage the *kernel-trick* to obtain the distances between the implicit feature-vector and the feature-vectors of the training-set.

Kernel-trick: Left-multiplying Eq. (3.22) with $\phi(\boldsymbol{y}_j)^\top$ and substituting for $\boldsymbol{U}_y$ from Eq. (3.15), we obtain $\forall j \in \{1, \dots, n\}$:

$$k_y(\boldsymbol{y}_j, \boldsymbol{y}_i) = \Big(\sum_{k=1}^{n} k_y(\boldsymbol{y}_j, \boldsymbol{y}_k)\boldsymbol{h}_k^\top\Big)\Big(\boldsymbol{\Lambda} - \sum_{l,m=1}^{n} \boldsymbol{h}_l k_y(\boldsymbol{y}_l, \boldsymbol{y}_m)\boldsymbol{h}_m^\top\Big)^{-1} \Big(\sum_{k=1}^{n} \boldsymbol{h}_k k(\boldsymbol{x}_k, \boldsymbol{x}_i)\Big).$$

The above could be reduced in the matrix form as follows:

$$\boldsymbol{k}_{\boldsymbol{y}}(\boldsymbol{y}_i) = \boldsymbol{K}(\mathcal{Y})\boldsymbol{H}^\top\Big(\boldsymbol{\Lambda} - \boldsymbol{H}\boldsymbol{K}(\mathcal{Y})\boldsymbol{H}^\top\Big)^{-1}\Big(\boldsymbol{H}\boldsymbol{k}_{\boldsymbol{x}}(\boldsymbol{x}_i)\Big), \tag{3.23}$$

where $\boldsymbol{k}_{\boldsymbol{y}}(\boldsymbol{y}_i) = [k_y(\boldsymbol{y}_1, \boldsymbol{y}_i), \dots, k_y(\boldsymbol{y}_n, \boldsymbol{y}_i)]^\top$. These distances in the feature space can be used in various *pre-image methods* to obtain the output $\boldsymbol{y}_i$. In the next section, we briefly discuss the pre-image problem and some common methods.

Pre-image Problem The advantage of using a kernel, $k(\boldsymbol{y}_t, \boldsymbol{y}_j) = \langle \phi(\boldsymbol{y}_i), \phi(\boldsymbol{y}_j)\rangle_{\mathcal{H}_y}$, is that all computations can be implicitly performed in feature space and the exact mapping $\psi : \mathcal{Y} \to \mathcal{H}_y$ is not required. However, this gives rise to the pre-image problem: given a point $\Psi \in \mathcal{H}$, find $\boldsymbol{y} \in \mathcal{Y}$ such that $\Psi = \psi(\boldsymbol{y})$. This pre-image problem is known to be ill-posed [128] as the pre-image might not exist [185] or, if it exists, it may not be unique. Instead, an optimization problem is formulated to find the approximate pre-image $\boldsymbol{y}^\star$:

$$\arg\min_{\boldsymbol{y}^\star} \|\Psi - \psi(\boldsymbol{y}^\star)\|^2 \tag{3.24}$$

While various techniques exist to approximately solve this problem [25, 94, 79, 225], we discuss two particular methods that can conveniently be incorporated into the RKM framework.

Kernel Smoother: A kernel smoother [187] is essentially a weighted-average of nearby data-points in the input space where both the weights and the nearby-points are given by the kernels in Eq. (3.23). Let $n_r < N$ be the number of nearest-neighbors, then the approximate pre-image for time-step i is given by:

$$\tilde{\boldsymbol{y}}_i = \sum_{j=1}^{n_r} \frac{k_y\left(\boldsymbol{y}_j, \boldsymbol{y}_i\right)\boldsymbol{y}_j}{\sum_{l=1}^{n_r} k_y\left(\boldsymbol{y}_l, \boldsymbol{y}_i\right)}.$$

With this method, the implicit assumption is that the forecasted point lives within the simplex of training data-points. Further, in the limit-case $n_r = 1$, only the nearest training point is chosen as the prediction. In our experiments, we treat n_r as the hyperparameter.

Kernel Ridge Regression: Alternatively, one can *learn* the pre-image map using kernel ridge regression [225]. Consider a specific case of Eq. (3.24) where the denoised point $\Psi = \boldsymbol{U}_y \boldsymbol{h}^\star$ (see Eq. (3.19)) i.e. $\arg\min_{\boldsymbol{x}^\star} \|\boldsymbol{U}_y \boldsymbol{h}^\star - \psi(\boldsymbol{y}^\star)\|^2$. Then one could formulate the following problem to learn the pre-image map $\Gamma \colon \mathcal{H} \mapsto \mathcal{Y}$:

$$\arg\min_{\Gamma} \sum_{i=1}^{n} \ell(\boldsymbol{y}_i, \boldsymbol{U}_y \boldsymbol{h}_i) + \lambda \Omega(\Gamma),$$

where $\ell(\cdot, \cdot)$ is a suitable loss function, Ω is the regularization function on the parameters Γ. Since $\mathcal{Y}$ is a vector space $\mathbb{R}^d$, we can consider the above as kernel ridge regression problem with kernel k. This gives the mapping $\Gamma_j(\boldsymbol{U}_y \boldsymbol{h}_i) = \sum_{r=1}^{n} \alpha_r^j k(\boldsymbol{U}_y \boldsymbol{h}_r, \boldsymbol{U}_y \boldsymbol{h}_i) = \sum_{r=1}^{n} \alpha_r^j k(\boldsymbol{h}_r, \boldsymbol{h}_i), \forall j \in \{1, \ldots, n\}$ where the coefficients are obtained by $\alpha = \left(\boldsymbol{K}^\top \boldsymbol{K} + \lambda \mathbb{I}\right)^{-1} \boldsymbol{K} \boldsymbol{Y}$.

Algorithm 3.1: MV-RKM hyperparameter tuning with grid-search

Input: Time series $\{\boldsymbol{x}_i\}_{i=1}^n, \boldsymbol{x}_i \in \mathbb{R}^d$
Output: Optimal kernel parameters

while *validation error* ≥ 0 **do**
- Init. kernels k_x, k_y; lag p ; `// Sample from grid`
- $\boldsymbol{X} \leftarrow \{[\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_{i-p}^{(1)}, \boldsymbol{x}_i^{(d)}, \ldots, \boldsymbol{x}_{i-p}^{(d)}]\}_{i=p+1}^{n-1}$; `// AR format`
- $\boldsymbol{Y} \leftarrow \{\boldsymbol{x}_{i+1}\}_{i=p+1}^{n-1}$; `// One-step ahead`
- $\boldsymbol{K}(\mathcal{X}) \leftarrow \boldsymbol{X}$, $\boldsymbol{K}(\mathcal{Y}) \leftarrow \boldsymbol{Y}$; `// kernel matrices`
- $\boldsymbol{H}, \boldsymbol{\Lambda} \leftarrow (\boldsymbol{K}(\mathcal{X}), \boldsymbol{K}(\mathcal{Y}))$; `// eigen-decomposition Eq. (3.16)`

Computational complexity

The eigenvalue decomposition for the training (Eq. (3.16)) requires slightly less than $\mathcal{O}(n^3)$ operations, since the matrices are hermitian. The complexity of the predictions in the latent space is $\mathcal{O}(n)$ (Eq. (3.21)), since it requires computation of kernels $\boldsymbol{k}_x(\boldsymbol{x}_i)$. Further, if k_y is a linear kernel, then the cost of prediction is constant, i.e. $\mathcal{O}(1)$ (Eq. (3.22)). Otherwise, one has to solve the pre-image problem and the complexity depends on the choice of pre-image method used. For instance, with the kernel-smoother, it is $\mathcal{O}(n_r)$ and with kernel ridge regression, it is $\mathcal{O}(n^3)$, since it requires matrix inversion.

Algorithm 3.2: MV-RKM recursive prediction

Input: Initial lagged vector: $\boldsymbol{x}_0 \in \mathbb{R}^{(p+1)d}$, time-steps n'
Output: Recursive forecasts $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{n'}\}$

```
while i ≤ n' do
    if i ≠ 0 then
        x_i ← {y_{i-1}, x_{i-1}, ..., x_{i-p}}
    Compute k_x(x_i)                          // similarity vector O(n)
    Get h_i (see Eq. (3.21))                  // latent vector O(1)
    if ψ(·) is known & invertible then
        y_i ← solve Eq. (3.22)                // O(1)
    else
        Compute k_y(y_i)                      // similarity vector O(n)
        y_i ← solve Eq. (3.24)                // pre-image problem
    i++
```

3.2.4 Experiments

In this section, we perform ablation studies on our proposed model on synthetic and real-world datasets to study the influence of different hyperparameters on the embeddings $\boldsymbol{H}$, which we refer to as latent components, and the predictions. Additionally, we show how visualizing the latent embeddings can help with explaining the data and model predictions. Finally, the model is quantitatively compared against other standard time series models. The source code used during the experiments can be found on github [6].

Ablation study

Sine wave. A simple sine wave is used to illustrate how the models learns the dynamics present in the input space, as latent representations $\boldsymbol{h}_i$. After hyperparameter tuning, we obtain a closed-loop attractor in the latent space (ref. Fig. 3.4). The forecasted embeddings of the unseen test set, indicated by the green curve, follow the same manifold as the embeddings of the training set as indicated with the blue curve. When the forecasted embeddings are transformed back to input space, it becomes apparent that the model has learned the data correctly, reflected by the high accuracy of the forecasted data points.

Sum of sines. A synthetic dataset was created as the sum of two sine waves as follows: $\sum_{i=1}^{n} \alpha_i \sin(2\pi\omega_i t + a_i)$, where $n = 2$, $\alpha_i \in \{1, 0.2\}$, $\omega_i \in \{1, 20\}$,

[6] https://github.com/hannesdm/multiview-kernel-pca-time-series

$a_i \in \{0, 0\}$. Using this dataset, we illustrate how the predictions of the model can be explained using the orthogonal latent components. On Fig. 3.5, the latent components are visualised on the right. On the left, the predictions of the model are shown where iteratively more and more of the plotted components are used. Visually, it can be seen that more components correspond with a more accurate forecast. This is further shown by plotting the error, in terms of the mean squared error, as done on Fig. 3.6a. When visualised, the orthogonal components offer a measure of explainability and show how the model decomposes the signal into uncorrelated components. Generally, we observe that higher frequency components are extracted for lower lag values and lower frequency components for high lag values (see first and second row in Fig. 3.5). Empirically, it was found that a lower lag means that a higher number of components are required to fit the model sufficiently well.

SantaFe. The same experiment was performed on the SantaFe data, which is a real world chaotic uni-variate time series dataset. We can observe the same effects as with the sine experiments in Fig. 3.5 (last row). The predictions improve given increasingly more of the visualised components. Further, the components can be observed to be able to capture the jumps in the chaotic time series, given a certain lag. It can be observed that, given a sufficiently large lag, more components correlate with a lower prediction error (see Fig. 3.6b). This effect is similar as in principal component analysis where only the top components are needed for a good reconstruction of the data.

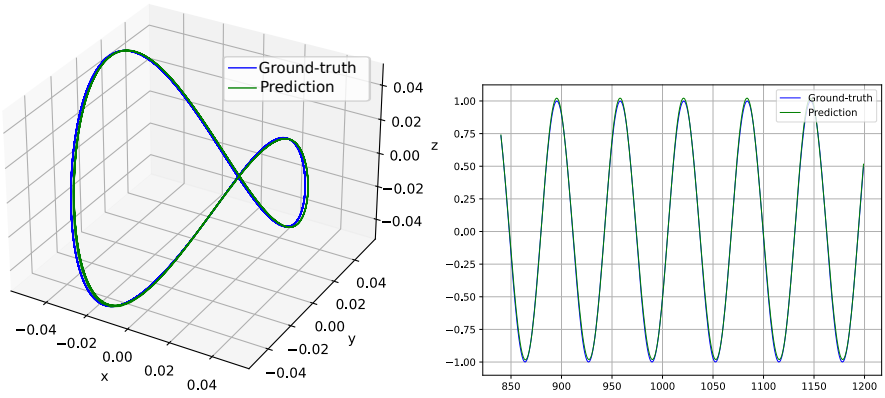


Figure 3.4: MV-RKM on sin wave. The embedded data points are *temporally* de-correlated in the embedding space.

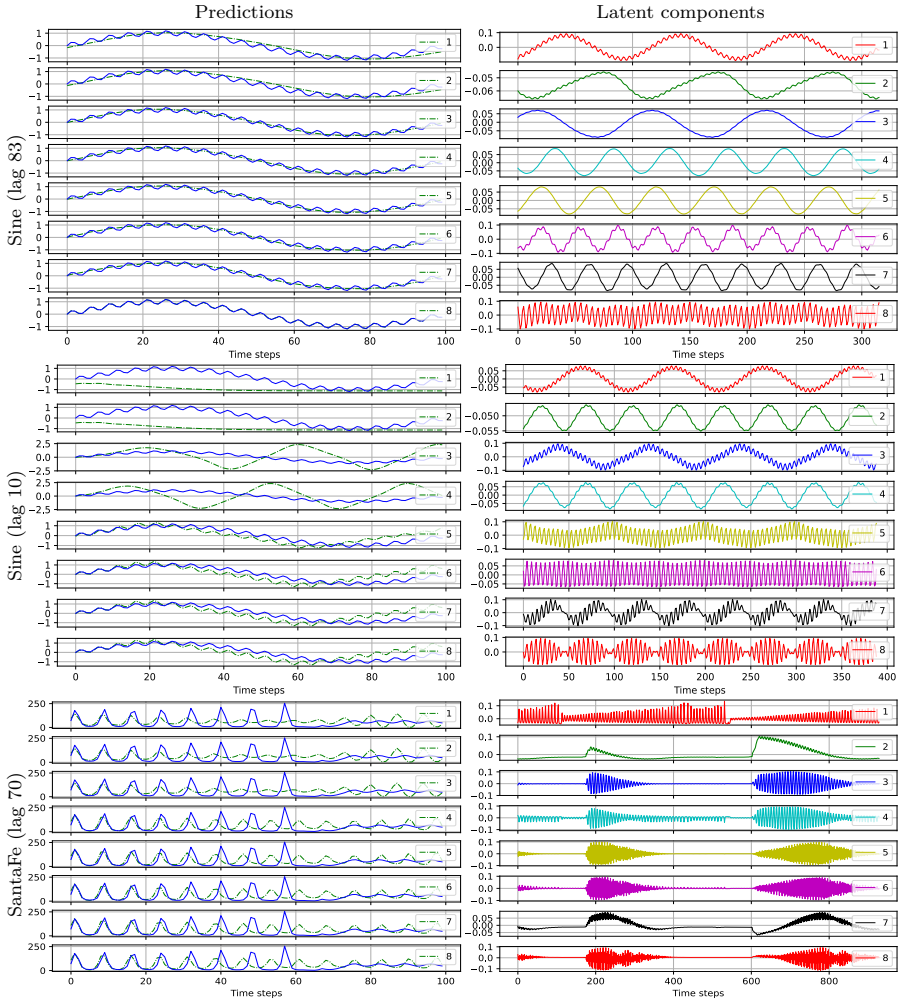
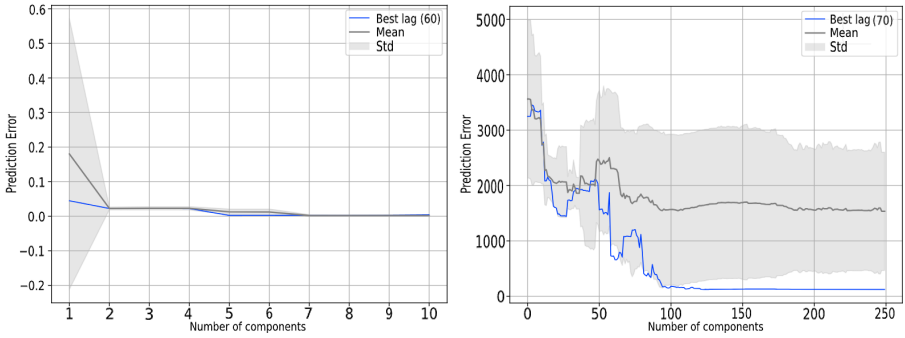


Figure 3.5: Predictions (left) and components (right) for the sum of sine and SantaFe experiments. More components correspond to a better prediction. A lower lag for the same parameters generally generate higher frequency components and more components are required for the same performance.

Real-world experiments

Baselines: To provide a benchmark study, we consider six baseline models: Recurrent Neural Networks (RNN), Long-Short Term Memory networks



(a) Sum of sine (b) SantaFe

Figure 3.6: Prediction error with different number of components on the sum of sines (left) and SantaFe (right) datasets. The blue curve indicates the prediction of the model for the best lag. The gray curve indicates the mean prediction error over multiple lag values with corresponding standard deviation. It can be seen that more components indicate a better reconstruction until a plateau is reached.

Table 3.2: Mean squared error on the forecasted data. Standard deviation for 10 iterations between brackets for the stochastic models.

Data	RNN	ARMA	KRLS	LSTM	LS-SVM	RRKM	RKM-MV	
							(k_y rbf)	(k_y lin)
SantaFe	3037.17 (±625.36)	2224.55	416.17	2272.13 (±328.87)	113.78	119.06	**90.23**	127.83
Lorenz	386.01 (±51.53)	412.28	273.00	331.49 (±61.30)	**131.87**	247.19	182.65	163.659
Chickenpox	34329.95 (±9513.07)	23571.35	19505.99	22296.24 (±3097.18)	15541.28	20716.91	**15479.93**	16807.56
Energy	16002.11 (±1809.89)	24797.40	11994.98	14781.65 (±3126.80)	10635.00	12764.097	**10385.07**	10474.06
Turbine	2401.12 (±644.53)	1317.67	1193.82	1817.38 (±552.21)	1226.87	1299.915	**1126.55**	1291.72

(LSTM), Autoregressive Moving Average (ARMA), Kernel-Recursive Least-Square [217] (KRLS), non-linear autoregressive formulation of the Least-Squares Support Vector Machines [199] (LS-SVM), Recurrent Restricted Kernel Machine [148] (RRKM). Further we consider two variants of the proposed model i.e. when $k_y(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle_{\mathscr{H}}$ and $k_y(\boldsymbol{y}_i, \boldsymbol{y}_j) = \exp(-\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2/2\sigma_y^2)$; with $k_x(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma_x^2)$ in both cases.

Datasets: We consider both uni-variate and multi-variate time series: two chaotic time series (SantaFe laser[7] [222], Lorenz attractor[8]) and three real world datasets (Chickenpox[9] [174], Belgian appliances energy consumption[10] [27] and

[7] `https://rdrr.io/cran/TSPred/man/SantaFe.A.html`
[8] `https://www.openml.org/search?type=data&sort=runs&id=42182&status=active`
[9] `https://archive.ics.uci.edu/ml/datasets/Hungarian+Chickenpox+Cases`
[10] `https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction`

Gas turbine[11] [86]). See C.2 for further details on these datasets.

On each dataset and method, hyperparameter tuning has been performed with grid-search and the result of the best set of parameters, quantified as the mean squared error, is shown in Table 3.2. For all methods, the entire validation set is forecasted, in *recursive* fashion, with the initial vector as the last point of the training set. The scores for RNN, ARMA and RRKM are taken from [148] since the experiment setting remains the same. From the table, we see that the proposed model RKM-MV outperforms the baselines on most datasets except Lorenz where the LS-SVM has lower error than RKM-MV. Also note that these two models are closest in performance, compared to the rest, since both are in NAR setting. However, the solution to LS-SVM requires solving a linear system (ref. C.3) in contrast with the eigenvalue decomposition of the proposed model. Another important difference is in the prediction schemes.

Moreover, LS-SVM outperforms RKM-MV when k_y is linear on most datasets except the Belgian appliances energy dataset. However, the ability to include non-linearity on k_y is an added advantage of the RKM-MV model which allows it to outperform other methods.

3.2.5 Conclusion

In this work, we introduced a time series forecasting model based on multi-view kernel PCA. This framework provides new insights into time series modeling such as latent space dynamics and novel relations between kernel PCA and time series forecasting. For future work, we believe a further extension of this model can be made to a multi-source setting. For example, when the data is coming from multiple weather stations and a forecast has to be made for all stations simultaneously. Custom kernels can be used to better capture the relationship between data-points. Moreover, existing techniques in kernel methods literature can be leveraged to improve the computational complexity, such as the Nyström approximation or random Fourier features. Besides the topics mentioned in this work, the framework can be further explored towards other tasks involving time series such as denoising, handling missing values and clustering.

[11] `https://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set`

Chapter 4

Unsupervised Embeddings for Categorical Variables

This chapter has been previously published as:

H. De Meulemeester and B. De Moor. "Unsupervised Embeddings for Categorical Variables". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207703

Contribution

Hannes De Meulemeester performed the research and wrote the paper.

Abstract

Real-world data sets often contain both continuous and categorical variables yet most popular machine learning methods cannot by default handle both data types. This creates the need for researchers to transform their data into a continuous format. When no prior information is available, the most widely applied methods are simple ones such as one-hot encoding. However, they ignore many possible sources of information, in particular, categorical dependencies, which could enrich the vector representations. We investigate the effect of natural language processing techniques for learning continuous word-vector representations on categorical variables. We show empirically that the learned vector representations of the categorical variables capture information about the variables themselves and their dependencies with other variables similar to how word embeddings capture semantic and syntactic information. We also show that machine learning models using unsupervised categorical embeddings are competitive with supervised embeddings, and outperform them when fine-tuned, on various classification benchmark data sets.

4.1 Introduction

Most machine learning models assume that their input values come from a continuous set (e.g. $\mathbb{R}^d$) or that, at least, a distance or similarity between inputs is defined. This is true for many applications but real-life data sets often contain both continuous and categorical variables. When presented with categorical values, it is often not obvious how to convert these to a continuous representation. Current categorical coding schemes may, very generally, be divided into nominal and ordinal.

Ordinal schemes, such as Helmert coding or polynomial coding [39, 224], assume an order and/or a structure in the variables. When such assumptions are not possible or sensible, the values may be treated as nominal and encoded using methods that replace the categorical values by certain calculated statistics, e.g. frequency encoding and target encoding [126]. Frequency encoding replaces the categorical value by its (normalized) frequency. Target encoding techniques replace the categorical values by certain statistics computed using the current variable value and their corresponding target variable values (e.g. mean encoding). The most popular technique, which does not calculate any statistics nor requires any assumptions, is one-hot encoding, also known as dummy encoding [91]. This technique replaces all values by equidistant indicator vectors. The disadvantage of this method is that the vector size increases with the cardinality of the variable. Furthermore, it does not capture

any useful information about the variable itself or its interaction with other variables. As such, this encoding scheme completely leaves the interpretation and transformation of the variables and its values to the machine learning model.
In natural language processing (NLP), however, the standard is to encode the input, such as words, as continuous vector representations, called embeddings [5]. Recently, there has been some interest in learning embeddings [70, 177] for general categorical variables instead of using the standard encoding techniques. However, the focus has been on learning embeddings as a part of the supervised classification model, limiting their reusability. To our knowledge, few researchers investigated the general applicability of unsupervised NLP embeddings methods to categorical features.
This paper is organized as follows. In section 4.2 embeddings in natural language processing and the most popular methods to create them are introduced. In section 4.3 we discuss how these embedding methods may be applied to categorical variables. In section 4.4 we show the ability of categorical embeddings to extract structure and dependencies and how they may improve classification performance compared to one-hot encoding and supervised neural embedding methods. We conclude the paper in section 4.5 and offer possible insights into future work in section 4.6.

4.2 Natural Language Processing

The data representation in natural language processing evolved from standard encoding techniques such as one-hot encoding and N-grams [83] to rich continuous vector space representations, called embeddings. Originally, neural embeddings were simply the continuous output of an intermediate neural network layer [16]. However, word embeddings only became widely accepted as the new standard after the introduction of the Skip-gram and Continuous Bag-of-words (CBOW) models [129]. These models allowed for efficient training due to their simple log-linear structure. Many extensions to these embeddings models have been made to allow them to work on i.a. character [87], sentence [106] or document [96] level.

The goal of embeddings is to capture useful semantic and syntactic information from the input data. One of the most noticeable properties about these embeddings is the ability to perform vector arithmetic in the embeddings space, that meaningfully translates back to the input space. The most famous example that illustrates this effect is that when one subtracts the vector for *Man* from *King* and adds instead the vector for *Woman*, one gets the vector for *Queen*: $vector(King) - vector(Man) + vector(Woman) \approx vector(Queen)$

[129]. It should be noted that recent work argued that this should be taken with a grain of salt [21, 137].
As an alternative to neural embeddings, we can use the matrix factorization methods. In general, matrix factorization or decomposition is the process of transforming a matrix into a multiplication of multiple, often structured, matrices with desired properties. In natural language processing, these methods do not work on the texts themselves, as the embedding methods, but on varying types of co-occurrence counts extracted from the corpus. Following the creation of such a co-occurrence matrix, the embeddings are created by applying low-rank matrix approximation methods, such as the Singular Value Decomposition (SVD). One of the most influential examples is Latent Semantic Analysis (LSA) [44], which works on the term-document matrix of a corpus. Another approach that instead uses a term-term co-occurrence matrix is the Hyperspace Analogue to Language (HAL) method [113]. However, a problem with count-based methods is that very frequent or infrequent words contribute a disproportionate amount. As such, recent work often focuses on how to properly transform the counts such that useful semantic information is extracted, while attempting to eliminate the side effects of these models [172, 26, 97].

4.2.1 CBOW and Skip-gram

The Continuous Bag-of-Words and Skip-gram models [129], commonly referred to as word2vec, are unsupervised methods for learning word vector representations. The CBOW model predicts a word based on the surrounding context words, while the skip-gram model predicts the context words based on the current word. Multiple improvements on these models have been made. It has been shown that for the Skip-gram model sub-sampling of frequent words and the use of negative sampling instead of hierarchical softmax allows for faster training and higher quality word vectors [130].
Words may also be split into multiple parts, N-grams, in order to capture subword information in N-gram embeddings. These embeddings may then be joined together in order to create the corresponding word embeddings [20, 68].

4.2.2 LSA

Latent Semantic Analysis (LSA) [44] produces word and documents vector representations using a term-document co-occurrence matrix. The rows of this matrix represent the terms and the columns represent the documents. Tf-idf [91] normalization is applied to the co-occurrence matrix to reduce the weight of uninformative high-frequency words. Finally, the dimensionality is reduced by

performing the (truncated) Singular Value Decomposition (SVD). The resulting low rank approximation is used to obtain the word embeddings.

4.2.3 GloVe

One of the limitations of local context methods, such as CBOW and Skip-gram, is that they do not take global statistics of the data into account. GloVe [155] is a method for creating word embeddings that combines the local context methods and the global factorization methods. The algorithm first constructs the word-word co-occurrence matrix X, denoting X_{ij} as the number of times j occurs in context of word i. Using this matrix, the goal is to discover the relations between words, formulated as the ratio of their co-occurrence probabilities in some context. It can be shown that this goal can be structured as a weighted least squares regression problem,

$$\min_{w,b} \sum_{i,j=1}^{V} f\left(X_{ij}\right) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2. \tag{4.1}$$

With $w_i, \tilde{w}_i \in \mathbb{R}^d$ and $b_i, \tilde{b}_j$ respectively the (context) word vectors and biases for the words in vocabulary V. The weighting function f has as goal to limit the effect of frequent (and infrequent) co-occurrences and was selected to be,

$$f(x) = \begin{cases} \left(x/x_{\max}\right)^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise.} \end{cases} \tag{4.2}$$

This function has two additional, positive-valued, parameters: $x_{\max}$ and α. The $x_{\max}$ determines at which point the function returns 1 and the α value is the power used to, possibly non-linearly, scale the fraction. It was found empirically that setting the values for $x_{\max}$ and α to be respectively 100 and 3/4 offered the best performance for natural language processing tasks [155].

4.3 Methodology

Although categorical variables are very closely related to words, two important differences must be considered. The first is that the interpretation of a combination of categorical variables does not depend on the location of those variables. Put differently, the columns of the input data matrix may be permuted without loss of meaning. This is not the case in natural language, where the order of characters, words and sentences are a large part of what determines

the meaning of the text.
The second difference lies in the relations between the values of the categorical variables. It is perfectly valid for a value, or even a complete category, to be independent or unrelated to the values of other variables. This is not the case in a meaningful sentence or text. No matter how unimportant a word, it is still needed to adhere to the syntax or to convey the correct meaning.
To translate these changes to the existing word embeddings methods, we need to make some adjustments. The first is the context window length. Since sentences have different lengths and the assumption may be made that words closer together are more strongly related than those further away, it makes sense to limit the size of the context window to some reasonable number. In the case of categorical variables, the window should initially span all variables since the length is always the same and the meaning of these variables is location independent.
In the case of GloVe, there are two parameters that require extra attention: the weighing of occurrences and the α value (cf. Eqn. 4.2). By default, the values in the co-occurrence matrix are weighted by the distance to the focus word. Based on this distance, the co-occurrence count of a context word is updated by $1/d$, with d the distance. In this way, occurrences that are further away are assumed to be less important than those closer. For categorical variables, this count should always be updated by one, independent of the distance.
The α parameter determines a transformation of the a co-occurrence counts, with a value 1 meaning the identity. Empirically, a value of 0.75 was shown to perform well on natural language. In our experiments, we found that for certain data sets values $\alpha \geq 1$ performed better, while for other data sets, values even lower than 0.75 were needed to achieve the best performance. This indicates the need to treat the α as a tuning parameter.
In the next section, we show how these changes to the embeddings methods allow them to be applied to categorical variables. We apply them to an artificially created data set to clearly show their respective strengths and then show that we can make the same observations on a more complicated data set. Finally, we show empirically that these embeddings offer an increase in classification performance compared to when using one-hot encoding and are competitive with supervised neural embedding methods.

Activity	Animal	Disposition
Barks	Dog	Playful
Roars	Cat	Lazy
Talks	Bird	Aggressive
Sings	Penguin	Calm
Dances	Mouse	

Relation	Probability
P(Dog | Barks)	0.8
P(Playful | Dog)	0.85
P(Lazy | Cat)	0.8

Table 4.1: Left: The categorical variables with their corresponding values of the artificial data set. Right: The corresponding manually created relations present in the data set.

4.4 Experiments

4.4.1 Simple Example

In this section we use a handcrafted example data set to illustrate how the embedding methods may be used to extract dependencies and structure out of categorical variables. The data set contains three categorical variables: Activity, Animal and Disposition. By default, all values of a variable are equally likely to appear. In addition to this, three relations, in the form of conditional probabilities have been created to simulate the dependency between variables. For example, given that the value *Barks* appears in the Activity column, the second column will contain *Dog* with a probability of 0.8.
On figure 4.1 the word vectors are depicted using t-Distributed Stochastic Neighbor Embedding (t-SNE) [117] as a dimensionality reduction and visualisation technique. Perplexity values of 5 to 10 often provided the clearest figures. The choice was made to use t-SNE for all embedding visualisations because this technique generates clearer and more interpretable figures than alternatives such as principal component analysis (PCA). However, it is important to note that, for all experiments, the exact same observations can be made on figures where PCA was used as the dimensionality reduction technique. To show this, the PCA visualisations of the two best models are additionally provided on figure 4.2. All three neural embedding methods were set up to generate embeddings of size 50. Additionally, for GloVe, the number of epochs and α were set to be 50 and 1.5, respectively. For LSA, the truncated SVD was performed using 2 components[1].

This simple example was created in order to show the relative strengths and

[1]The code for the methods can be found at https://github.com/hannesdm/unsupervised-categorical-embeddings/

weaknesses of the studied methods. As can be seen on figure 4.1, both CBOW and Skip-gram appear to perform well in discovering which values belong to which variables, visualised by the distinct clusters. The methods are less effective at discovering the relations between the values themselves: they often fail to detect more difficult relations such as the one with Dog, Barks and Playful. Skip-gram performs slightly better in that aspect.
Making the link with natural language processing, one could say that these methods are better at distinguishing syntax over semantic relations.
LSA is able to extract the relations between the values of different categorical variables, but is often worse at differentiating between the categorical variables themselves. This is indicated by the less clearly defined clusters of variables.
True to its original goal, GloVe performs equally well at extracting relations and recognizing different categorical variables while respecting the distances between the embedded variable clusters.
In general GloVe performs the best at extracting the structure and relations out of this simple data set. Although, it should be said that all techniques are able to represent more information than would be the case when using popular transformations such as one-hot encoding.

4.4.2 Nursery Data

The UCI nursery data[2] [52] is a data set containing only categorical variables. It is extracted from a hierarchical decision model originally developed to rank applications for nursery schools [139]. The structure derived from a decision model makes it a suitable candidate to illustrate the use and advantages of categorical embeddings on a more complicated data set. The data consists of 12 960 samples, with eight categorical variables that represent the hierarchical structure (see table 4.2) and a categorical variable that represents the final decision of the model.
To compare with what a standard statistical method is able to extract, Cramer's V score [35, 17] for all attribute pairs was calculated. It is a measure of dependency between nominal variables with 0 indicating no dependency and 1 indicating complete dependency. The score was 0 for all variable pairs except for the pairs including the target variable, which are presented in table 4.2. These scores show that a strong relation exists between the 7th attribute (the health conditions) and the target variable. However, the test does not detect any other relations, such as the hierarchical structure.

The results of the embeddings methods can be found in figure 4.3. As in the simple example, CBOW and Skip-gram perform exceedingly well at creating

[2]https://archive.ics.uci.edu/ml/datasets/nursery

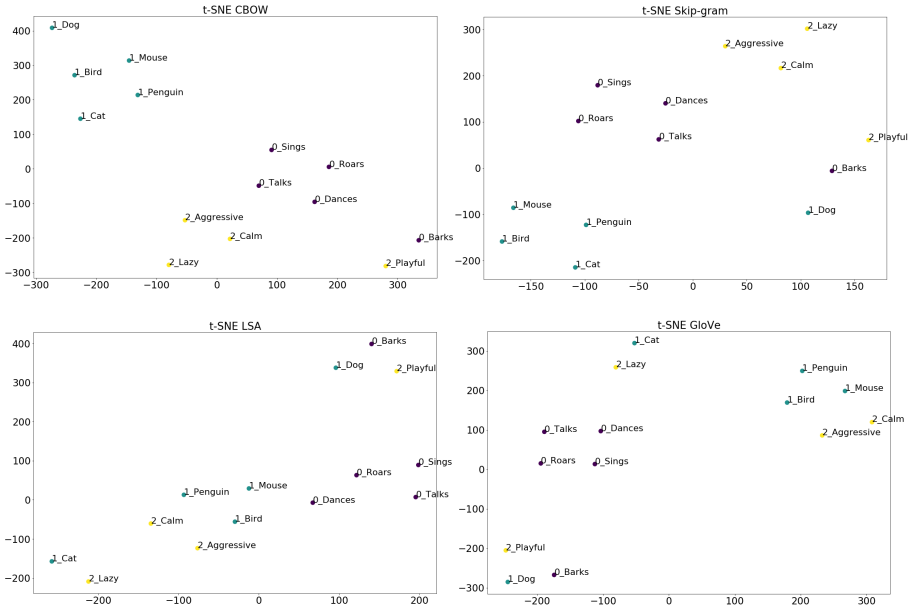


Figure 4.1: The t-SNE visualisations of respectively CBOW, Skip-gram, LSA and GloVe for the simple example. CBOW and Skip-gram perform well at representing the categorical variables, shown by the clear variable clusters. LSA is able to extract all dependencies, as indicated by the closeness of the related values in embedding space, but has less distinct clusters. GloVe performs well at both tasks.

representations of the categorical variables themselves. To a lesser degree, the strong relation between the health conditions and the target variable (variables 7 and 8, respectively) can also be observed.
LSA performs well at picking up the relation between variables 7 and 8 but has trouble identifying the structure present in the data. GloVe again shows the best results; in fact the method is able to almost fully capture the hierarchical structure of the decision model. The categorical variables which are aggregated together in the tree structure are located close in the embedding space. For example, variables 0-1 and 2-3 are aggregated into "Employment of parents and child's nursery" and "Family structure" respectively and are clustered together in the embedding space. The relationship extracted by Cramer's V, between variable 7 and 8, is also captured, demonstrated by the closeness of their corresponding embeddings.
As in the simple example described in the previous section, GloVe is able to extract the most information out of the data set. Additionally, the experiments

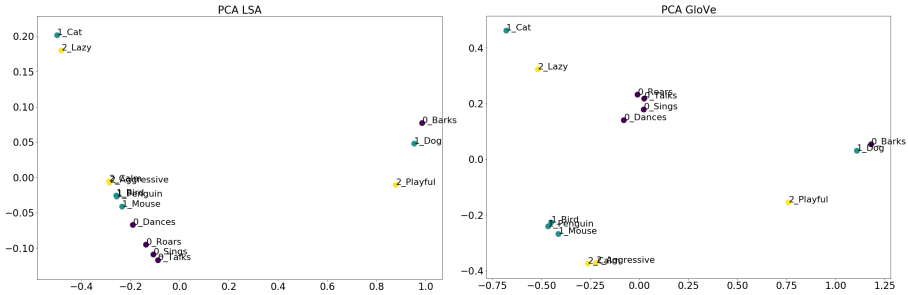


Figure 4.2: The PCA visualisations of the two best performing models, LSA and GloVe, for the simple example. The visualisations show the same relations as those that can be observed on the t-SNE plots: Both LSA and GloVe are able to extract the relations but GloVe is better at constructing clusters of the categorical values.

```
8: Final evaluation of applications for nursery schools
. Employment of parents and child's nursery
. . 0: Parents' occupation
. . 1: Child's nursery
. Family structure and financial standings
. . Family structure
. . . 2: Form of the family
. . . 3: Number of children
. . 4: Housing conditions
. . 5: Financial standing of the family
. Social and health picture of the family
. . 6: Social conditions
. . 7: Health conditions
```

Attribute	Cramer's V
0	0.214
1	0.246
2	0.047
3	0.072
4	0.112
5	0.075
6	0.106
7	0.731

Table 4.2: Left: Hierarchical structure of the variables of the nursery data. Right: The Cramer's V score, a measure for association between categorical variables, for the corresponding variables with regards to the final evaluation.

show that even on a more difficult, real, data set, all methods are able to capture the structure and dependencies, albeit to varying degrees.

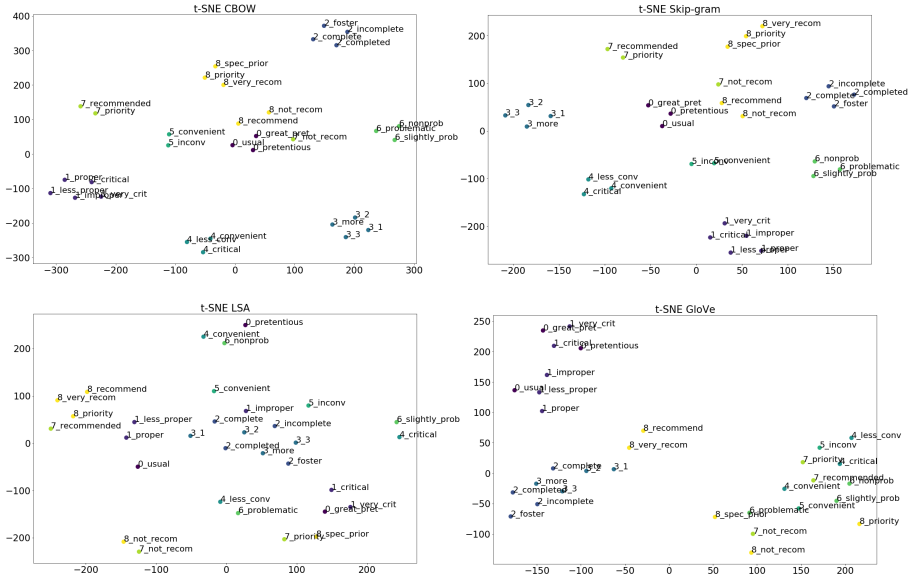


Figure 4.3: The t-SNE visualisations of respectively CBOW, Skip-gram, LSA and GloVe for the nursery data set. CBOW and Skip-gram do well at representing the categorical variables, indicated by their corresponding distinct clusters. LSA is able to extract more relations than CBOW and Skip-gram: some attributes of the same hierarchical level are closer together in the embeddings space. GloVe performs well at both.

4.4.3 Classification Performance

We compare the performance of the embeddings methods with the one-hot encoding baseline on the Breast cancer[3], CMC[4], Adult[5] and Credit[6] data sets from the UCI repository. The first two data sets do not contain any continuous features and are completely transformed using one-hot encoding or one of the embedding methods. For the last two data sets, the categorical variables are embedded, and the continuous features are left as is. For the CMC data set, the two minority classes are treated as a single class to create a binary classification task.

Classification models. All experiments were performed using the same two simple models: a logistic regression and neural network model, implemented

[3]https://archive.ics.uci.edu/ml/datasets/breast+cancer
[4]https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice
[5]https://archive.ics.uci.edu/ml/datasets/Adult
[6]https://archive.ics.uci.edu/ml/datasets/credit+approval

Method	Breast cancer				CMC			
	LR		NN		LR		NN	
	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best
One-hot	62.03 ± 1.61	64.42	70.38 ± 1.36	71.75	73.76 ± 1.41	76.12	82.80 ± 0.77	83.97
CBOW	66.86 ± 0.87	68.08	**71.96** ± 0.99	73.51	74.15 ± 1.72	77.47	85.16 ± 0.66	86.57
Skip-gram	**67.19** ± 1.57	**68.75**	71.76 ± 1.46	73.04	76.20 ± 1.12	78.33	84.93 ± 0.53	85.91
LSA	65.30 ± 1.12	67.77	71.70± 0.94	72.91	77.92 ± 1.08	80.32	84.33 ± 0.74	85.60
GloVe	**67.19** ± 1.14	68.30	71.50 ± 1.26	**73.78**	**79.10** ± 0.91	**80.58**	**86.05** ± 0.36	**86.68**
	Adult				Credit			
	LR		NN		LR		NN	
	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best
One-hot	81.88 ± 0.0	81.88	90.30 ± 0.28	90.60	86.85 ± 0.35	87.67	92.50 ± 0.17	92.71
CBOW	81.98 ± 0.0	81.98	90.49 ± 0.23	90.73	86.71 ± 0.41	87.68	92.37 ± 0.31	93.06
Skip-gram	81.98 ± 0.0	81.98	90.45 ± 0.22	90.79	86.72 ± 0.40	87.98	92.32 ± 0.35	92.81
LSA	**82.03** ± 0.0	**82.03**	90.41 ± 0.45	90.81	86.92 ± 0.35	87.72	**93.05** ± 0.20	**93.29**
GloVe	81.92 ± 0.0	81.92	**90.58** ± 0.37	**90.96**	**87.22** ± 0.36	**88.22**	92.66 ± 0.24	93.01

Table 4.3: Overview of the classification performance of the baseline one-hot encoding and the categorical embedding methods on the four UCI data sets. The reported values are the average AUC with standard deviation and the best observed AUC for a logistic regression model (LR) and a neural network with one hidden layer (NN). The best performing model is consistently one with as input categorical embeddings.

Architecture	Breast cancer		CMC		Adult		Credit	
	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best
One Hidden Layer	68.78 ± 1.52	71.78	83.25 ± 0.80	84.68	90.69 ± 0.13	90.88	92.25 ± 0.28	92.70
Two Hidden Layers	68.75 ± 1.34	71.52	83.25 ± 0.85	85.07	90.74 ± 0.20	90.99	92.50 ± 0.26	92.89
One Hidden Layer (Pretrained)	71.56 ± 1.33	74.19	**86.38** ± 0.37	87.22	91.08 ± 0.03	91.12	93.14 ± 0.19	93.43
Two Hidden Layers (Pretrained)	**72.30** ± 1.50	**74.81**	86.19 ± 0.72	**87.64**	**91.11** ± 0.06	**91.18**	**93.42** ± 0.26	**93.87**

Table 4.4: Overview of the classification performance of the neural embeddings with and without unsupervised pretraining. The reported values are the average AUC with standard deviation and the best observed AUC. For the Breast cancer, CMC and Credit data set the unsupervised methods outperform the neural embeddings without pretraining. In all cases, pretrained neural embeddings with fine-tuning outperform the standard supervised neural embeddings and achieve the best result on all data sets.

using respectively scikit-learn [154] and Tensorflow [120]. The neural network consists of a single hidden layer with 512 neurons, a RELu activation function and a dropout layer with a rate of 0.1 as regularization. The neural embedding architecture has a separate input, followed by an embedding layer, for each categorical variable of which the outputs are then concatenated. These concatenated embeddings are then used as input for one or more hidden layers. The experiments have been performed with one hidden layer and a second hidden layer as a reference to reflect the original architecture [70]. No architecture tuning has been performed as the goal is to compare the embeddings methods

as unbiased as possible. For the logistic regression model, the SAGA optimizer [46] was used while the neural network model was trained using the Adam optimizer [89]. For the embeddings methods, the embeddings size and the frequency scaling parameter, the α, were manually tuned. All models have been trained for at least 10 times. The average and best performance of the models, measured by the area under the ROC curve (AUC), is given in table 4.3. For the Adult data set, the AUC was calculated on the provided test data set and for the other three the 5-fold cross-validation score is reported.

Only categorical variables. For the Breast cancer and CMC data sets, all embeddings methods significantly outperform one-hot encoding. On the breast cancer data, Skip-gram and CBOW performed the best on average, with Skip-gram and GloVe obtaining the best performance for the logistic regression and the neural network model respectively. The best results were observed using an embedding size of 50 to 100 and an $\alpha <= 0.75$. For LSA, the number of components was set to be 3 to 5.

On the CMC data set, GloVe is the best performing embedding method consistently for both models. The optimal embedding size and α values were found to be respectively 20 to 50 and 0.25. For LSA, we achieved the best results using 5 components.

Mixed variables. For the two data sets containing both continuous and categorical variables, Adult and Credit, categorical embeddings again lead to the best results. However, the performance gain by the embeddings methods was less pronounced. On the Adult data, LSA performed best in combination with the logistic regression model, while GloVe outperformed the other methods when used in the neural network model. For both models Skip-gram and CBOW benefited from an embedding size of 20 and GloVe performed better with an embedding size of 100 and an α value of 0.75 or lower. For both models, 7 components were selected for the LSA embeddings.

On the credit data set, CBOW and and Skip-gram perform slightly worse on average than one-hot encoding. Nevertheless, the higher variability in their performance allows their best achieved results to outperform the baseline. The best performing embedding methods are GloVe and LSA for the logistic regression and neural network models respectively. These results were obtained with embedding sizes of 25 for the Skip-gram and CBOW models. GloVe performed better with an α value of 0.4 and an embedding size of 50-100. For the LSA method, 6 components were selected.

In general, the best performing model is consistently one with categorical embeddings as input, indicating that the discussed embeddings methods are able to extract useful information, such as structure and dependencies, that are helpful for the classification process. Among the embeddings methods, there is no technique that consistently outperforms the others. However, for the majority of our experiments GloVe, more frequently, offered the largest

classification performance increase. This, in combination with the visualisation experiments, where it was shown that GloVe was the best at i.a. extracting relations, makes it the preferred initial choice to create unsupervised categorical embeddings.

Supervised neural embeddings. The performance of supervised neural embeddings and unsupervised embeddings with fine-tuning is reported in table 4.4. The pretraining is performed using the best reported method in table 4.3, which is GloVe for all data sets except for the Credit data set where LSA performed better. For the standard supervised embeddings (without pretraining), small embedding sizes of 5-20 performed better than large sizes. The supervised neural embeddings also perform better than one-hot encoding. However, even without fine-tuning, the unsupervised embeddings outperformed the supervised embeddings on all data sets except on the Adult data. Finally, the unsupervised embeddings with fine-tuning achieve the best results on all data sets.

We explain the competitive, and often better, performance of the unsupervised methods by their ability to take global statistics (e.g. LSA, GloVe) into account. In addition, it is known that unsupervised pretraining, as in the case of the NLP embeddings, is beneficial to the performance and generalization capabilities of neural network models [56]. The experiments show that this is also the case for categorical variables.

4.5 Conclusion

In this work we studied the applicability of unsupervised NLP embedding methods to categorical variables. We observed that the learned continuous vectors do not only represent the categorical variables efficiently, but also capture dependencies and structure in an unsupervised way. Drawing the link with natural language processing, the ability for word embeddings to capture semantic and syntactic relations translates well to categorical variables.

Additionally, we have empirically shown that pretrained categorical embeddings outperform one-hot encoding on various popular classification benchmark data sets. Out of the discussed embedding methods, GloVe offered the most consistent improvements.

When comparing the unsupervised methods to the supervised neural embeddings, it was shown that they are at least competitive with each other. But more often than not, the unsupervised embeddings outperformed the supervised neural embeddings, especially when the data set consisted only of categorical variables. Finally, we showed that fine-tuned unsupervised embeddings consistently outperformed any other embedding method.

4.6 Future Work

Categorical embeddings provide an attractive alternative to commonly used methods such as one-hot encoding. However, for the NLP techniques to be applicable to categorical variables, some changes to the treatment of the context window and distance weighting needed to be made. Additionally, it was often necessary to tune the frequency weighting parameter α in order to obtain the best results. This offers future research directions into specialized categorical methods and weighting schemes. Additionally, real life data often contains both categorical and continuous data while the current unsupervised embedding methods only take the categorical part into account. Techniques may be developed that allow the training of the categorical embeddings to take these continuous values into account as well.

Chapter 5

Explainable Unsupervised Anomaly Detection for Healthcare Insurance Data

This chapter is under review and is available as a preprint as:

H. De Meulemeester, F. De Smet, J. van Dorst, E. Derroitte, and B. De Moor. "Explainable Unsupervised Anomaly Detection for Healthcare Insurance Data". Internal Report 23-129, ESAT-STADIUS, KU Leuven (Leuven, Belgium), 2023.

Contribution

Hannes De Meulemeester developed the workflow, implemented the methods used, performed the experiments and wrote the manuscript.

Abstract

Background: Waste and fraud are important problems for health insurers to deal with. With the advent of big data, these insurers are looking more and more towards data mining and machine learning methods to help in detecting waste and fraud. However, labeled data is costly and difficult to acquire as it requires expert investigators and known care providers with atypical behavior.

Methods: In this work we show how recent advances in machine learning can be used to set up a workflow that can aid investigators in discovering practitioners or groups of practitioners with unusual resource use in order to more efficiently combat waste and fraud. We combine three different techniques, which have not been used in the context of healthcare insurance anomaly detection: categorical embeddings to deal with high-cardinality categorical variables, state-of-the-art unsupervised anomaly detection techniques to detect anomalies and Shapley additive explanations (SHAP) to explain the model output.

Results: The method has been evaluated on providers with a known anomalous profile and with the help of experts of the largest health insurance fund in Belgium. The quantitative experiments show that categorical embeddings offer a significant improvement compared to standard methods and that the state-of-the-art unsupervised anomaly detection techniques generally show an improvement over traditional methods. In a practical setting, the proposed workflow with SHAP was able to detect a previously unknown, anomalous trend among general practitioners.

Conclusions: The proposed workflow is able to detect known care providers with atypical behaviour and helps expert investigators in making informed decisions concerning possible fraud or overconsumption in the health insurance field.

5.1 Background

Healthcare and, in turn, health insurance providers are a vital pillar of a modern welfare state. However, healthcare comes at an ever increasing cost. The European Union currently spends approximately 1.5 trillion euro yearly on healthcare which, on average, corresponds to 10.9% of GDP per country in 2020 [57]. In the United States, healthcare spending reached 4.3 trillion dollars [29] which is 18.3% of its GDP. The increased standard of care that is present in modern society means having access to the latest technological advances in medicine and care, which come at a premium [119].

It is no wonder that all stakeholders involved are looking to find areas of improvement such that healthcare spending can be decreased without compromising on quality of care. One of such areas is the detection of waste and fraud. In general, waste or fraud are almost never considered on its own, but as part of a greater whole containing waste, errors, abuse, fraud and corruption [53]. The main difference between these types of infringements is the intention and degree of severity. Only in the case of abuse, fraud and corruption can it be said that there is an intent to do wrong by finding loopholes or stretching a rule (abuse) or by actively and intentionally breaking the law (fraud and corruption). In the case of errors, a rule is unintentionally broken. Similar to this is waste or overconsumption, which includes care that is unnecessary or needlessly expensive. When simply looking at the billing data, as available to healthcare insurers, it can be difficult to differentiate between these types of infringements and care providers with valid resource use. Therefore, it is common to either group the categories together and look for general anomalies. Alternatively, specific, thematic, investigations can be performed to investigate known types of fraud.

In this work, we focus on the perspective of health insurers. Health insurance funds in Belgium are co-managers of the health insurance system and have a vested interest in managing this system as optimal as possible. Additionally, these health insurers are the closest to the source as they are the first to receive detailed billing data, making them the first line of defense.

However, waste and fraud are one of the most difficult problems for health insurers to deal with. Therefore, they are looking for new ways to help them with this task. With the advent of big data, these insurers are looking more and more towards data mining and machine learning methods to help in detecting anomalies in resource use data. Investigating waste or fraud in healthcare practitioners in principle requires a business specialist to go over the resource use records of all the care providers and to subsequently interview the practitioners with a potential anomalous profile. Such manual investigations are costly, inefficient and require a lot of time without tools to guide the investigator. Such tools can help by pre-selecting the care providers with a profile that could have a higher risk of inappropriate resource use. Manual investigations are also required to acquire labeled data if one wants to apply supervised machine learning techniques. This need for expert investigators to label the training data makes supervised methods more difficult. Additionally, supervised problems are limited in that only previously known types of waste and fraud are considered. As a result, the labeled data can contain additional false negatives with respect to inappropriate resource use in general. Finally, it is usually desirable to be able to identify new types of anomalies that warrant further investigation.

This research has been performed at the Christian Health Insurance Fund

(CM) [202]. The CM, which has over 4.5 million members, is the largest Belgian mutual health insurer and currently processes over 400 million expenditure records every year.

Contribution This work shows how a number of recent advances in machine learning can be used to improve anomaly detection on health insurance resource use data and to help healthcare experts make informed decisions about possible fraud or overconsumption in the field. We combine three different state-of-the-art techniques, which have not been used in the context of healthcare insurance anomaly detection previously, to create a novel workflow. Categorical embeddings [40] are employed to deal with the large number of categorical features and values. These embeddings allow for a more efficient representation of the data, which in turn speeds up machine learning methods and significantly increases their performance. A large variety of unsupervised anomaly detection techniques, each with their own unique advantages, are used during the experiments. We consider both classic methods, which have been a staple in the anomaly detection field, and novel methods, which have not been employed in the context of health insurance data. Finally, we illustrate how Shapley additive explanations (SHAP)[114] can be used to help experts identify anomalies and novel patterns in the data.

5.1.1 Related Work

Advanced statistical, data mining and machine learning techniques are a mainstay for fraud detection in many different areas [156, 2]. However, healthcare insurers are slower to implement these advanced types of techniques due to complex regulations, large decentralized data systems and high costs of commercial fraud detection packages[213] with questionable or at least unproven efficacy. Further, preparing the data and labeled data for machine learning is costly and manually intensive. Finally, investigating anomalies and subsequently, if necessary, deal with care providers with inappropriate resource use is a difficult and long process. Notwithstanding the slow integration, there has been some research into the field of unsupervised fraud detection in healthcare.

Chandola et al.[31] showed how text mining, social network analysis and temporal analysis can be applied to big healthcare data. For text mining, they performed topic modeling using latent Dirichlet allocation and determined that certain diagnosis topics were more likely to be related to fraudulent claims. Concerning social network analysis, graphs were created with the providers as nodes. They found that another fraudulent provider was present in 40% of the cases within 2-hops from a known fraudulent provider. Their temporal analysis

showed that anomalies could be detected using temporal features. They also showed that features of a temporal nature are useful as input to supervised models such as logistic regression classifiers.

Thornton et al.[204] used k-means clustering, using custom metrics to perform anomaly analysis. The metrics were developed in cooperation with experts and incorporated features such as visit length, patient retention, referral rate and visit frequency. The domain knowledge used is common in their research and they conclude that this expertise is vital in order to develop healthcare fraud detection techniques [205].

Van Capelleveen et al. [215] performed experiments on dentist Medicaid data and showed that previously unknown patterns can be detected using simple unsupervised methods such as linear models and k-means clustering. The authors worked closely together with field experts and concluded that their methods could be beneficial for a decision support.

Bauder et al. [12] show that it is possible to classify healthcare professionals into their respective fields based on billing data. The practitioners for which the models have difficulty classifying them are flagged as outlying. The authors also developed a novel univariate probabilistic programming anomaly detection method which provided interpretability on the feature level [11]. The method was later extended with a MARS regression model [13, 59]. Bauder et al. additionally employed traditional unsupervised anomaly detection models where it was shown that Local Outlier Factor [23] performed the best [14].

From the existing research, it can be seen that machine learning and data mining techniques can be beneficial when used in the field of unsupervised healthcare insurance anomaly detection. However, limited research has been performed in cooperation with European health care insurers and fewer still has been performed with the latest anomaly detection and explainability techniques.

5.2 Methods

In Figure 5.1 a general overview of the workflow can be found. First, the data is preprocessed such that a suitable data set is created for the machine learning models. Next, a wide variety of unsupervised anomaly detection techniques is employed to detect and rank the actual anomalous profiles. For the anomalous samples that have been detected, SHAP is employed such that for each practitioner the feature attribution, or importance, scores are calculated. Finally, the results are summarized and visualized to help the expert investigators in interpreting the results and guiding the investigation by helping

the experts to efficiently select the care provider profiles that might warrant further investigation. When employing the workflow, it was found useful to have an expert in the loop at all times such that the process can be further optimized. For example, it might turn out that a certain variable, that was included in the preprocessing, is not useful to determine anomalies or has an unwanted impact on the anomaly detection methods.

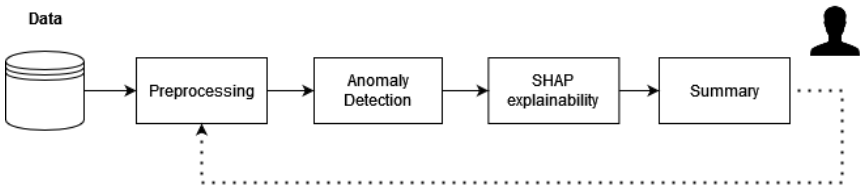


Figure 5.1: The general overview of the workflow to help expert investigators in detecting anomalous behavior and to pre-select care providers with a profile that could have a higher risk of inappropriate resource use. The health insurance data is first filtered and preprocessed to obtain a single, rich, feature vector for each practitioner. Next, anomaly detection is performed by a number of traditional and state-of-the-art anomaly detection techniques. The model output, for a sample to be investigated, is explained by using SHAP. Finally, the results are summarized in the form of visualisations and summary statistics to the expert investigator.

5.2.1 Preprocessing

Figure 5.2 shows a schematic overview of the preprocessing steps performed before employing the actual anomaly detection techniques. Resident experts were involved in the entire process, including the preprocessing, in order to identify useful features and interpret the data correctly. This allowed us to reduce the number of available features from hundreds to 66 different features that were identified to be potentially useful in characterizing anomalous behaviour.

Typically, tabular data contains both real and categorical data. Most machine learning methods however cannot by default be applied to categorical features. Currently, the standard is to use one-hot encoding, also known as dummy encoding [82], for categorical data. For a single categorical variable, such encoding is performed as follows. Consider a categorical variable C with n unique categories $\{c_1, c_2 \ldots, c_n\}$. Then a category c_i is transformed into a vector $x \in \mathbb{R}^n$ where each element is defined by equation 5.1.

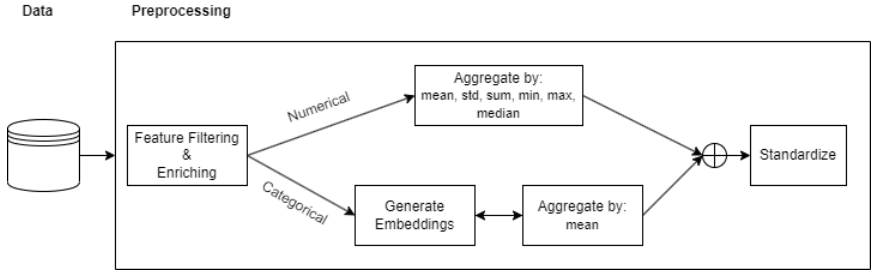


Figure 5.2: The preprocessing performed before performing anomaly detection. In close cooperation with experts, the features are filtered and enriched (e.g. using expert knowledge) such that a suitable set of the data and features are extracted that are useful in determining atypical behavior. Numerical features are aggregated per practitioner by their mean, standard deviation, sum, minimum, maximum and mean statistics. The categorical features are embedded using a categorical embedding method and aggregated using the mean. The data is then recombined and standardized by subtracting the mean and dividing by the standard deviation.

$$x_i = \begin{cases} 1 & \text{if } C = c_i \\ 0 & \text{if } C \neq c_i \end{cases} \tag{5.1}$$

However, for categorical features with a high cardinality this results in long feature vectors which greatly increases the dimensionality of the data set. Additionally, not all machine learning methods are able to efficiently handle high-dimensional inputs which can be detrimental to their performance. A better alternative is to transform categorical features by using categorical embeddings [40]. These embeddings have been shown to extract useful information, such as relations between the categorical variables, and result in a much smaller feature size. It was decided to use Latent Semantic Analysis (LSA) [44, 45] as the embedding method as this was shown to be the most consistent [40].

LSA originates from the information retrieval field where it is used to create word and document vector representations. For categorical variables, it was shown that a similar process can be used to create vector representations of the categorical values. First, a co-occurrence matrix X is constructed with as rows all possible categories (from all categorical variables) and as columns all actions (of all practitioners). Then, a binary matrix is constructed such that each entry of the matrix indicates whether a specific categorical value is present in a practitioner action or not. At this point, each categorical value

is represented in X by a large binary vector which could be used for machine learning tasks. However, this vector representation has as dimensionality p, the number of practitioner actions (rows) in the data set, which is often too large for any practical use. Therefore, in LSA, a rank-k approximation is made of X by means of the Singular Value Decomposition (SVD) as in Figure 5.3. This approximation extracts the most important relations from the co-occurrence matrix while removing redundancy and noise. The categorical embeddings for all categories are obtained by considering only the category part of the decomposition: $XP = CS \in \mathbb{R}^{c \times k}$ where each row corresponds with the embedding of a category with dimension k. Finally, the resulting embeddings are l2-normalized. This normalization is typical in information retrieval as it allows, for example, to easily compare two vectors by using the cosine distance by a dot product [9].

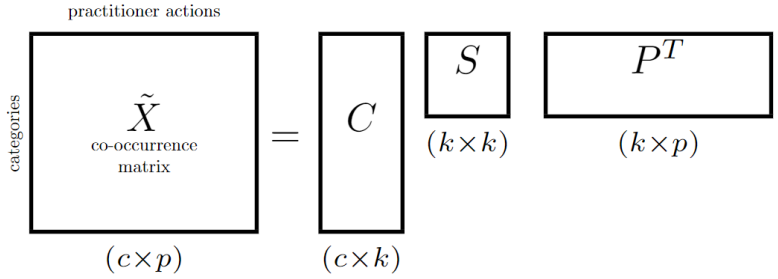


Figure 5.3: The Singular Value Decomposition step of LSA applied to categorical variables. The co-occurrence matrix, relates the categories with the data entries (the practitioner actions), gets decomposed into a rank-k approximation of the category space C, the practitioner actions space P and weights S which relates the two.

The embedding size k is a tuning parameter as the optimal value depends on the problem. Generally, a trade-off is made between the size and expressivity of the vectors. Larger embeddings can capture more information, up to a point, but make the learning process slower due to the larger dimensionality. In natural language processing, large embedding sizes (e.g. > 100) are usually preferred [236] while for categorical embeddings, smaller sizes already achieve a significant performance increase [40].

For the embedding size, it was decided to use 3 dimensions as this showed to provide a good performance improvement during our experiments while keeping the overall dimensionality manageable. Increasing the dimensionality offered little to no performance improvement while only increasing the computational complexity. The process described above creates an embedding, in our case a 3-dimensional vector, for each categorical value possible. Each value in the

original data set is then replaced (embedded) by the corresponding embedding. After embedding the categorical features, the data set contains only real-valued features.

Healthcare insurance data is time series data where each sample is an action for a particular practitioner or institution. Because of the different possible actions, varying lengths and the scale of the data, the time series are typically aggregated into a single vector per practitioner. The aggregation follows previous work [74, 14] such that all numerical features are aggregated by their sum, mean, median, standard deviation, minimum and maximum statistics. For the embedded data, the features are aggregated using the mean. Finally, the resulting feature columns are standardized before training by, for each feature, subtracting the mean and dividing by the variance.

5.2.2 Models

The experiments are performed using a wide variety of anomaly detection methods to investigate which types of algorithms are most suitable. This section gives a short overview of these methods.

Nearest neighbor methods are commonly used methods due to their simplicity and ease to explain. These methods take into account the k nearest neighbors of a data point to calculate their outlyingness score. In the simplest case, for the K Nearest Neighbors algorithm, the distance between a sample and its k nearest neighbors can be calculated and averaged. This average distance can then be used to rank the samples such that anomalies get a higher outlyingness score because they are furthest from their neighbors. A disadvantage of these methods is that they are generally less effective for high-dimensional problems as the effect of any single feature becomes less important when calculating the distance between two samples. K Nearest Neighbors (KNN) [163] and Local Outlier Factor (LOF) [23] have been successfully used on Medicare data [14] where these methods were the best performers. Additionally, we consider Angle-based Outlier Detection (ABOD) [93] and Subspace Outlier Detection (SOD) [92] which have been shown to work better on high-dimensional data. The final nearest neighbor method that is employed is Learnable Unified Neighborhood-based Anomaly Ranking (LUNAR) [66], a state-of-the-art local anomaly detection technique. This model combines the nearest neighbor approach with graph neural networks.

Isolation based methods take the entire data set into account and attempt to partition the data into regions such that small outlying groups are easily isolated from the rest of the data. These methods generally employ decision trees, which makes them explainable and fast because they are easily parallizable. Isolation

Forest (IForest or IF) [110] use an ensemble of binary decision trees constructed from a random sample of the data. The anomalies are isolated by randomly splitting the data at each level based on a random feature. An outlyingness score is calculated based on the length of the path down the decision trees of a sample. Unsupervised Random Forest (URF) [1] uses the standard random forest in an unsupervised setting by introducing synthetic data. The proximity matrix provided by the random forest algorithm is then used to calculate an anomaly score. A more recent method in this category is called Isolation-based Nearest-Neighbor Ensembles (INNE) [10]. It was shown that this method outperforms Isolation Forest models when it comes to detecting local anomalies and high-dimensional data with many irrelevant features.

One-class methods attempt to learn the single class that is represented by the training data. Typically, such methods attempt to find a suitable trade-off between a hypersphere that encompasses as much of the training data as possible while at the same time keeping the hypersphere small. The most well-known of these methods is the One-Class Support Vector Machine (OCSVM) [182]. Deep Support Vector Data Description (DeepSVDD) [176] can be seen as an extension to the OCSVM where the encompassing hypersphere is learned together with non-linear feature representations by using a deep neural network.

Distribution estimation methods attempt to learn the data distribution, or an approximation thereof, which can then be used to identify anomalies. A Variational Autoencoder (VAE) is a probabilistic generative model in the form of an autoencoder. The reconstruction probability of a trained VAE can be used as an anomaly score. Lightweight On-line Detector of Anomalies (LODA) [157] is a method where the data set is projected on a number of random vectors of which an ensemble of one-dimensional histograms is then created. The method takes inspiration from ensemble and boosting methods and shows that a collection of weak models can achieve state-of-the-art results while being one of the fastest anomaly detection methods. Copula-Based Outlier Detection (COPOD) [104] and Empirical-Cumulative-distribution-based Outlier Detection (ECOD) [105] are two recent parameter-free methods. The methods use different ways to inspect, for each dimension, the tails of their univariate empirical cumulative distribution function and aggregate this information into an anomaly score.

5.2.3 Explainability

Machine learning approaches have been slow to be adapted for use in the healthcare insurance field. One important reason for this is that it can be difficult to interpret and explain the model output. Understanding why a machine learning model makes a prediction is a requirement if it is to be used

by experts in order to make a decision concerning the presence of fraud or overconsumption.

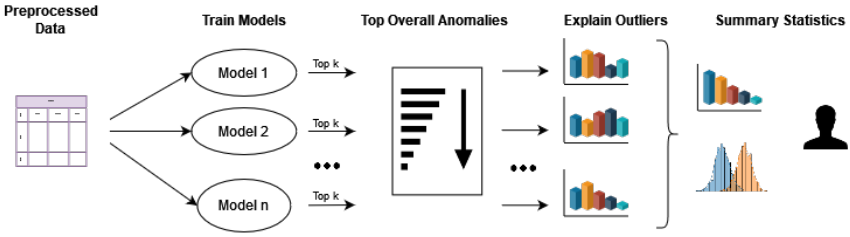


Figure 5.4: The process for explainable anomaly detection. Preprocessed data is provided to n anomaly detection models. The top k anomalies for each models are explained using SHAP. Next, the anomalies are ranked based on how many times they are detected by the anomaly detection methods. finally, The explanations in combination with standard statistics are summarized and provided to an expert to help them in their decision making.

Recently, a model-agnostic technique, called SHAP [114], was proposed that allows any model output to be explained. The method can assign an importance value to any feature for a single prediction or a group of predictions by using Shapley values [189], originating from cooperative game theory. Calculating SHAP values is computationally costly except when using linear or tree-based models [114]. However, it is not necessary to calculate these values for the entire data set as only the top outlying samples are of interest. Figure 5.4 shows how SHAP is incorporated in the workflow. For the trained anomaly detection models, the top k outlying care providers are selected. Each of these samples may be considered separately and investigated individually. However, to further limit the search window, we found it informative to sort the anomalies based on the number of times it is present in the top k for all models. As a result, an expert is provided a list of profiles with unusual resource use where the most anomalous, over all models, are ranked higher.

A large advantage of SHAP is that Shapley values are fully additive, this means that the respective SHAP values of multiple features can be added to get their corresponding total importance. This is especially useful when working with embeddings, or feature statistics, where a single feature is transformed into a new representation with a larger dimensionality. In such cases, it is often not easy to calculate an importance score for the original feature using traditional methods whereas using SHAP we simply need to sum the scores for all dimensions. This process is illustrated in Figure 5.5. The left graph shows the feature contribution of the categorical embeddings and numerical feature statistics. For both categorical and numerical features, the dimensionality of

the input features is larger than that of the original features. The feature contributions of all new dimensions are summed to find the total importance of a feature with regards to predicting the anomaly score of a practitioner.

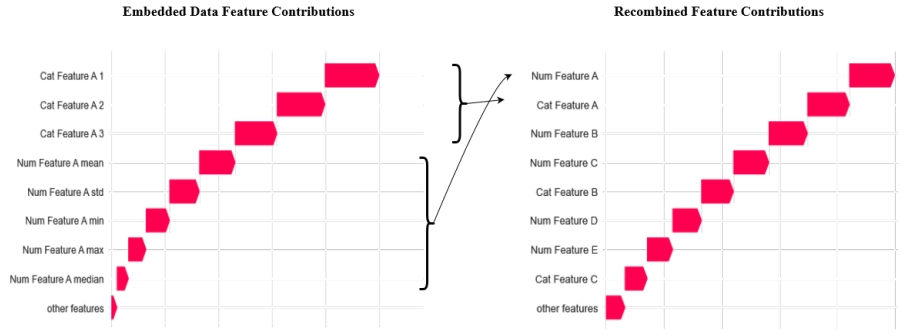


Figure 5.5: The SHAP feature contributions for the feature embeddings are summed such that the contribution of the original feature is achieved. For a numerical feature, all values are aggregated per practitioner using the mean, standard deviated, minimum, maximum and median. For a categorical variable, the values are embedded and aggregated by the mean. Both methods expand a feature into multiple dimensions which means there are as many feature contributions (importance scores) per original feature. SHAP has the advantage that these scores can be summed to achieve the overall importance score for the original feature.

At this point, the top anomalous samples over all models are listed together with their SHAP feature contributions. The investigator is able to see why a model assigns a high anomaly score to a practitioner. This is useful as different techniques might flag a sample as outlying for different reasons. However, a single explanation for all models provides a better overview, especially when working with many different models. Therefore, all SHAP scores are rescaled and the mean over all models is calculated to provide a single plot that shows the feature contributions over all models. Finally, it was found to be useful to provide additional statistics from the original time series data. Specifically, for an outlying sample, the SHAP values of the top the aggregated features are shown together with the univariate feature distributions of the time series data for the top outlying features. For the categorical features, the distribution of the practitioner is shown together with that of the population. For the numerical features, the values are compared with the average population values and standard univarate statistics such as the Z-score are shown.

One important feature, that is always provided to the expert in the form

of summary statistics, is the nomenclature for the action performed by the institution or practitioner. For example, in the United States, these are the HCPCS codes [34]. In Belgium, these codes are simply referred to as nomenclature codes [169].

The final result is a clear overview where a field expert can immediately identify why a sample is flagged as outlying and whether further investigation for the sample is needed. An example of an importance plot with univariate statistics for a categorical and numerical feature are shown in figure 5.6. The top anomalous practitioners are investigated by first inspecting the overall SHAP importance plot. The expert will then investigate the summary statistics of the most important features in combination with the nomenclature codes in order to identify the nature of the anomaly. The proposed workflow filters and sorts the anomalous practitioners, reducing the search window, and provides an explanation for the detection.

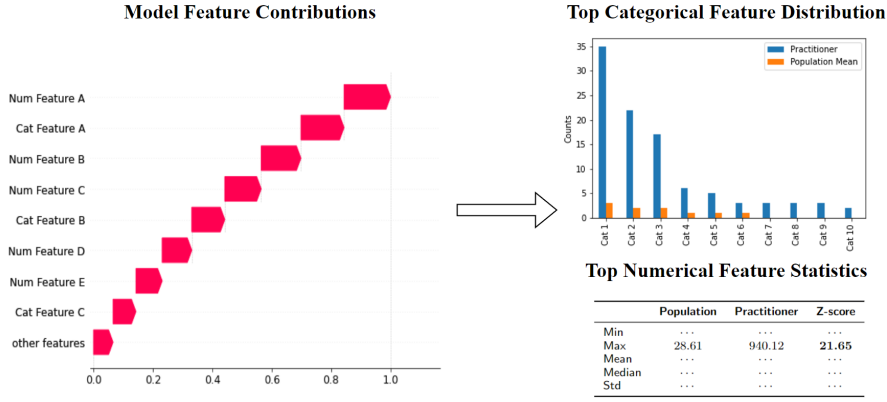


Figure 5.6: The SHAP model feature contributions, for a single sample, indicate which feature contribute most to the model output i.e. the anomaly score. The features contributing most to the sample being considered anomalous are then further investigated. For a categorical feature, the distribution of the categorical values are shown and compared to the average of all practitioners. For a numerical feature, the aggregate features (i.e. the minimum, maximum, mean, median and standard deviation) are shown, compared with the mean of the all practitioners and the Z-score is provided. This provides a clear overview for the expert investigator that shows why a practitioner's behavior is flagged as anomalous.

5.3 Results and Discussion

5.3.1 Belgian Health Insurance Data

The CM data analyzed in this paper has only been used for research purposes, as a proof of concept, without any operational goals. All research has been performed on pseudonymized data at the CM under the supervision of the chief medical officer. No re-idenfitication of the data points has been performed at any time, regardless of whether the samples have been identified as anomalous or not.

In this section, we consider the data set of all general practitioners (GPs) in Belgium that have more than 50 entries per year in the CM database. This results in 11851 different GPs. A set of 38 general practitioners was provided who were known for their anomalous profiles due to excessive use of certain billing codes. This setting is a decidedly imbalanced task where only 0.3% of the data set are identified as known anomalies. It is important to note that there could be, and likely are, more unknown anomalies present in the data. The original data contains 12899707 individual medical acts, in the form of resource use data, of all general practitioners. Each practitioner action has 66 features of which 54 contain categorical features and 12 contain numerical features. The cardinality for the categorical features varies between 2 and 159155. After the preprocessing, described in the previous section, the data sets have the final number of features and samples as shown in Table 5.1.

Table 5.1: The number of samples and features of the data sets used during the experiments. The original data is aggregated and transformed into one-hot encoded and embedding data sets.

Data	**Samples**	**Features**
Original	12899707	66
One-hot	11851	191707
Embeddings	11851	234

5.3.2 Experiment Results

The performance of the anomaly detection methods is quantified using the area under the ROC curve (AUROC). All models output a real-valued anomaly score which is used to sort the samples with more anomalous samples being ranked

higher. The ROC curve is calculated using these anomaly scores and labels indicating whether the practitioner has a known anomalous profile.

The AUROC has two advantages which makes it suitable to the discussed setting. It is immune to class imbalance, which is certainly the case, and it assigns a higher score when the outlying class is ranked higher than the other class. This second point makes it especially suitable as one goal is to investigate which anomaly detection methods can best help experts with identifying novel and outlying samples, which would be the methods that are able to rank anomalies higher.

The experiments were performed using Python where the models have been implemented using the Scikit-learn[154], PyOD [242] and Tensorflow [120] packages. In table 5.2 the results of the experiments on the transformed CM data sets can be found.

A clear difference can be observed between the one-hot encoded and the embedding data. For almost all methods, the embedding method significantly outperforms the current standard, one-hot encoding. Healthcare insurance data sets are rich in categorical features with often thousands of different values. One-hot encoding makes the number of features explode in size while not providing any additional useful information which is detrimental for the performance and time complexity of most machine learning methods. The only exception to this observation is the LUNAR method. The learnability aspect of the graph neural network approach is able to overcome the limitation of standard neighbor methods where their performance tends to deteriorate for very high-dimensional data.

Overall, the best performing methods are VAE and LODA with INNE, OCSVM and COPOD following closely behind with the proposed embedding method. It can be seen that the more recent and state-of-the-art anomaly detection methods generally outperform the methods such as KNN and isolation forest which are traditionally used for health insurance data. The exception to this is the one-class SVM which is one of the best performing methods on the embedding data.

The experiments show that unsupervised anomaly detection methods can help to detect or prioritize the same anomalies that have been detected by experts. However, to do so effectively, special care must be taken during the preprocessing step, particularly with regards to the categorical features.

Table 5.2: Area under the ROC curve of for the one-hot and embedding encoded CM general practitioner data for the following algorithms: Local Outlier Factor (LOF), K Nearest Neighbors (KNN), Isolation Forest (IF), Unsupervised Random Forest (URF), Angle-base Outlier Detection (ABOD), Empirical-Cumulative-distribution-based Outlier Detection (ECOD), Copula-Based Outlier Detection (COPOD), Variational Autoencoder (VAE), Histogram-based Outlier Score (HBOS), Isolation-based Nearest-Neighbor Ensembles (INNE), One-Class Support Vector Machine (OCSVM), Deep Support Vector Data Description (DeepSVDD), Subspace Outlier Detection (SOD), Learnable Unified Neighborhood-based Anomaly Ranking (LUNAR) and Lightweight Online Detector of Anomalies (LODA).
For the stochastic algorithms, the performance shown is the average with standard deviation over 10 different runs. The embedding method offers a clear performance increase for all methods. VAE and LODA achieve the best performance with INNE, OCSVM and COPOD following closely behind.

Model	**Onehot**	**Embeddings**
LOF	0.60	0.69
KNN	0.60	0.78
IF	0.53 (±0.04)	0.76 (±0.03)
URF	0.51 (±0.01)	0.62 (±0.01)
ABOD	0.60	0.76
ECOD	0.60	0.75
COPOD	0.60	0.81
VAE	0.51 (±0.00)	**0.82 (±0.00)**
HBOS	0.51	0.79
INNE	0.52 (±0.01)	0.80 (±0.01)
OCSVM	0.60	0.81
DeepSVDD	0.58 (±0.02)	0.61 (±0.03)
SOD	0.61	0.77
LUNAR	0.77 (±0.01)	0.78 (±0.01)
LODA	0.56 (±0.01)	**0.82 (±0.01)**

5.3.3 Explainability

In this section, we show how unsupervised machine learning methods in combination with SHAP can be used to aid decision making and help experts in discovering new types of anomalies. This will be done by considering a real-life example of a novel trend that was discovered while performing the experiments.

When investigating the top anomalous profiles of general practitioners, a particular practitioner had high SHAP feature contribution scores for the features indicating the discipline, location, subcategory and the total billing amount of the medical intervention. When investigating the provided univariate feature statistics, the billing amount was significantly higher than is standard for general practitioners. The feature distribution for the discipline showed that most actions were part of the orthopedics discipline. The type of the action showed a large amount of actions related to surgeries. The location indicated that these took place at general hospitals. Finally, the distribution of nomenclature codes revealed actions such as arthroplasty of the knee or hip using a prosthesis, meniscectomy etc. As a summary, this investigation into the practitioner showed that they mainly aided in surgeries involving orthopedics in a hospital.

During further investigation, multiple practitioners were identified by the algorithm with similar behavior, indicating a trend which was previously unknown to experts at the CM. While this behavior by itself does not indicate fraud or waste, it is clearly outlying for a standard general practitioner and, as such, provides novel avenues of investigation for experts. Important for the experts was that the results of the used anomaly detection techniques were interpretable: the investigator was able to immediately identify the anomalous behavior using the combination of SHAP, simple univariate statistics and visualisations.

5.4 Conclusion

In this work, we discussed a workflow for explainable anomaly detection in the healthcare insurance field. The workflow implements and applies a number of recent advances in machine learning: categorical embeddings, state-of-the-art anomaly detection methods and SHAP explainability, which are novel to unsupervised anomaly detection in healthcare insurance. The experiments show that unsupervised anomaly detection techniques can successfully detect and prioritize practitioners with known anomalous behavior which were provided by experts. It was shown that the current standard of one-hot encoding categorical features is detrimental for most models due to the high cardinality of the categorical features, which are common in healthcare insurance data. Instead, we propose to use categorical embeddings, which provide a significant increase in performance.
Finally, we showed how the machine learning techniques can be used in combination with SHAP and simple summary statistics to explain the predicted anomalies and aid expert investigators. We illustrated how the approach was

successfully used in the field by providing an example of a previously unknown anomalous trend for general practitioners

Chapter 6

Conclusion

Contributions

In this thesis, generative adversarial networks, restricted kernel machines and healthcare insurance outlier detection have been discussed. Specifically, various unsupervised representation learning methods have been proposed to increase performance, improve diversity and explainability of the considered machine learning topics. In the following, an overview of all contributions per chapter is provided.

In Chapter 2, diversity in generative adversarial networks was considered. A novel type of GAN was proposed, named BuresGAN, which includes an additional term in the loss function based on the Bures distance between the generated and real batch distributions. This term promotes the matching of the diversity of the data generated by the generator and the diversity on the original dataset. It was experimentally demonstrated that the proposed method is able to successfully combat mode collapse and improves the quality of the generated samples. In addition, ridge leverage score sampling was introduced for training GANs in the context of complete mode coverage. It was shown how three different feature maps can be used to calculate the scores: an implicit feature map, a fixed explicit feature map when prior knowledge is available and the discriminator as a feature map when no prior knowledge is available. Experimentally, it was shown that RLS sampling allows the GAN to learn all modes more equally and is beneficial for the quality of the samples when minority modes are present in the data.

In Chapter 3, two methods for time-series modeling and forecasting were

proposed for restricted kernel machines. For the first method, it was proposed to add an additional term to the RKM objective function, which maximizes the correlation between the past and current latent variables. This allows the model to capture the dynamics of the data set in the latent space and create an autoregressive forecasting model starting from the latent space. For the second method, it was proposed to treat time-series forecasting and modeling as a multi-view problem where the lagged input and the 1-step ahead prediction are treated as different views. It was shown how both methods can be trained by solving a multi-view KPCA problem and discussed multiple pre-image methods, which can be used to map the forecasted point from the latent space to the input space. Finally, the performance of the methods was tested on several well-known time-series benchmark datasets.

In Chapter 4, unsupervised categorical embeddings were introduced. It was shown how embedding methods originating from natural language processing can be applied to categorical variables with minimal changes. Experimentally, it was confirmed that the embeddings are able to extract information about the categorical variables themselves and their relation with others. Additionally, it was demonstrated that the unsupervised categorical embeddings provide a performance increase when used as input for machine learning models on various classification benchmark datasets.

In Chapter 5, a complete novel workflow was proposed for anomaly detection in healthcare insurance data. The unique problems that are present in this field were addressed starting from the data processing up to the explainability of the used methods. A novel method to transform categorical variables, unsupervised categorical variables, was introduced in order to deal with the large amount of highly informative categorical data that is present in healthcare insurance data. To perform the anomaly detection experiments, a combination of classical anomaly detection methods and novel state-of-the-art methods were implemented. Finally, to help experts make informed decisions, SHAP was added to the workflow to explain the anomaly detection models. The complete workflow and its results were successfully validated in cooperation with experts from the CM and a previously unknown anomalous trend was discovered.

Future Work

In the final section of this thesis, a number of future research directions are discussed.

BuresGAN was proposed, which augments a standard GAN, that minimizes the KL-divergence between real and generated data, with the Bures distance between

generated and real batches. This shows that it is possible and advantageous to combine loss functions based on the Wasserstein distance and the KL-divergence. It would be interesting to investigate other ways on how these two different types of learning signals could be combined.

For restricted kernel machines, a novel extension for time-series data was introduced. However, state-of-the-art time series models are specialized deep learning models with multiple layers and various ways of combining and connecting the different components. Similarly, more complicated architectures can be created using RKMs. For example, it was observed that different input lags for the time series model result in different components being extracted. Therefore, it would be interesting to create a model with multiple views corresponding to different lag sizes.

The proposed unsupervised categorical embeddings are efficient real-valued representations that can increase the performance of machine learning methods. One disadvantage of the method is that only categorical variables of the dataset are taken into account when learning the embeddings. This can limit the amount of information that the embeddings can capture if there are few categorical variables or if the categorical variables themselves do not contain a lot of information useful for the machine learning problem. An interesting research direction is to take all variables into account when learning categorical embeddings.

In order to help experts with detecting anomalies in healthcare insurance data, a completely unsupervised workflow was developed. It was shown how the workflow is able to succesfully detect anomalous patterns and help explain the output of the models. However, because of its completely unsupervised nature, it can be difficult to guide the anomaly detection algorithms to detect specific patterns or anomalies. Future work can include methods that incorporate prior knowledge into the workflow to look for specific patterns. Additionally, it is possible for the unsupervised anomaly detection algorithms to detect anomalous patterns because they are rare and not because they indicate fraud or overconsumption. It would be useful to exclude known valid patterns from the workflow such that only novel anomalous patterns are detected.

Appendix A

The Bures metric for generative adversarial networks

A.1 Details of the theoretical results

Let A and B be symmetric and positive semi-definite matrices. Let $A^{1/2} = U\,\mathrm{diag}(\sqrt{\lambda})U^\top$ where U and λ are obtained thanks to the eigenvalue decomposition $A = U\,\mathrm{diag}(\lambda)U^\top$. We show here that the Bures distance between A and B is

$$\mathcal{B}\left(A,B\right)^2 = \min_{U\in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 = \mathrm{Tr}(A+B-2(A^{\frac{1}{2}}BA^{\frac{1}{2}})^{\frac{1}{2}}), \quad \text{(A.1)}$$

where $O(\ell)$ is the set of $\ell \times \ell$ orthonormal matrices. We can simplify the above expression as follows

$$\min_{U\in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 = \mathrm{Tr}(A) + \mathrm{Tr}(B) - 2\max_{U\in O(\ell)} \mathrm{Tr}(A^{1/2}B^{1/2}U) \quad \text{(A.2)}$$

since $\mathrm{Tr}(U^\top B^{1/2}A^{1/2}) = \mathrm{Tr}(A^{1/2}B^{1/2}U)$. Let the characteristic function of the set of orthonormal matrices be $f(U) = \chi_{O(\ell)}(U)$ that is, $f(U) = 0$ if $U \in O(\ell)$ and $+\infty$ otherwise.

Lemma A.1.1. *The Fenchel conjugate of $f(U) = \chi_{O(\ell)}(U)$ is $f^\star(M) = \|M\|_\star$, where the nuclear norm is $\|M\|_\star = \mathrm{Tr}(\sqrt{M^\top M})$ and $U, M \in \mathbb{R}^{\ell\times\ell}$.*

Proof. The definition of the Fenchel conjugate with respect to the Frobenius inner product gives

$$f^\star(M) = \sup_{U \in \mathbb{R}^{\ell \times \ell}} \operatorname{Tr}(U^\top M) - f(U) = \max_{U \in O(\ell)} \operatorname{Tr}(U^\top M).$$

Next we decompose M as follows: $M = W\Sigma V^\top$, where $W, V \in O(\ell)$ are orthogonal matrices and Σ is a $\ell \times \ell$ diagonal matrix with non negative diagonal entries, such that $MM^\top = W\Sigma^2 W^\top$ and $M^\top M = V\Sigma^2 V^\top$. Notice that the non zero diagonal entries of Σ are the singular values of M. Then,

$$\max_{U \in O(\ell)} \operatorname{Tr}(U^\top M) = \max_{U \in O(\ell)} \operatorname{Tr}(W\Sigma V^\top U) = \max_{U' \in O(\ell)} \operatorname{Tr}(\Sigma U'),$$

where we renamed $U' = V^\top U W$. Next, we remark that $\operatorname{Tr}(\Sigma U') = \operatorname{Tr}(\Sigma \operatorname{diag}(U'))$. Since by construction, Σ is diagonal with non negative entries the maximum is attained at $U' = \mathbb{I}$. Then, the optimal objective is $\operatorname{Tr}(\Sigma) = \operatorname{Tr}(\sqrt{M^\top M})$. □

By taking $M = A^{1/2}B^{1/2}$ we obtain (A.1). Notice that the role of A and B can be exchanged in (A.1) since U is orthogonal.

A.2 Additional Experiments

A.2.1 Alternating training

In the sequel, we also provide the results obtained thanks to Alt-BuresGAN, described in Algorithm A.1, which is parameter-free and yield competitive results with BuresGAN. Nevertheless, the training time of Alt-BuresGAN is larger compared to BuresGAN.

Algorithm A.1: Alt-BuresGAN

Sample a real and fake batch ;
Update G by minimizing $\mathcal{B}(\hat{\boldsymbol{C}}_d, \hat{\boldsymbol{C}}_g)^2$;
Update G by minimizing V_G;
Update D by maximizing $-V_D$;

A.2.2 Another matrix norm

In this section, we also compare BuresGAN with a similar construction using another matrix norm. Precisely, we also use the squared Frobenius norm

(Frobenius^{2}GAN), i.e., with the following generator loss $V_G + \lambda \|C_d - C_g\|_F^2$, with $\lambda = 1$. We observe that

- Frobenius^{2}GAN also performs well on simple image data sets such as Stacked MNIST,
- Frobenius^{2}GAN has a worse performance with respect to BuresGAN on CIFAR and STL-10 data sets (see Table A.5).

This observation emphasizes that the use of the Bures metric is requisite in order to obtain high quality samples for higher dimensional image data sets such as CIFAR and STL-10 and that the simpler variant based on the Frobenius norm cannot achieve similar result.

A.2.3 Timings

The timings per iteration for the experiments presented in the paper are listed in Table A.1. Times are given for all the methods considered, although some methods do not always generate meaningful images for all datasets. The methods are timed for 50 iterations after the first 5 iterations, which is used as a warmup period, following which the average number of iterations per second is computed. The fastest method is the vanilla GAN. BuresGAN has a similar computation cost as GDPP. We observe that (alt-)BuresGAN is significantly faster compared to WGAN-GP. In order to obtain reliable timings, these results were obtained on the same GPU Nvidia Quadro P4000, although, for convenience, the experiments on these image datasets were executed on a machines equipped with different GPUs.

A.2.4 Best inception scores achieved with DCGAN architecture

The inception scores for the best trained models are listed in Table A.2. For the CIFAR datasets, the largest inception score is significantly better than the mean for UnrolledGAN and VEEGAN. This is the same for GAN and GDPP on the STL-10 dataset, where the methods often converge to bad results. Only the proposed methods are capable of consistently generating high quality images over all datasets.

Table A.1: Average time per iteration in seconds for the convolutional architecture. Average over 5 runs, with std in parenthesis. The batch size is 256. For Stacked MNIST, we use a discriminator architecture with 3 convolutional layers (see Table A.12). For readability, the computation times for the Grid and Ring dataset are multiplied by 10^{-3}.

	Grid	Ring	stacked MNIST	CIFAR-10	CIFAR-100	STL-10
GAN	**1.28**(0.008)	**1.28**(0.016)	**0.54**(0.0005)	**0.65**(0.02)	**0.64**(0.0008)	**6.00**(0.01)
WGAN-GP	9.0(0.04)	8.96(0.04)	2.99(0.004)	3.41(0.009)	3.41(0.006)	36.5(0.03)
UnrolledGAN	3.92(0.008)	3.92(0.028)	1.90(0.002)	2.17(0.003)	2.18(0.004)	21.99(0.06)
MDGAN-v2	3.24(0.008)	3.24(0.032)	1.66(0.002)	1.98(0.002)	1.98(0.002)	18(0.03)
VEEGAN	1.96(0.012)	1.96(0.02)	0.56(0.006)	0.66 (0.006)	0.65(0.004)	6.10(0.03)
GDPP	15.68(1.0)	16.16(0.2)	0.69(0.02)	0.80(0.02)	0.80(0.02)	7.46(0.03)
PacGAN2	1.72(0.016)	1.72(0.024)	0.77(0.006)	0.91(0.007)	0.91(0.007)	8.02(0.008)
Frobenius^{2}GAN	2.32(0.024)	2.28(0.028)	0.98(0.004)	1.09(0.004)	1.09(0.003)	13.6(0.01)
BuresGAN	11.16(0.016)	11.92(0.04)	0.72(0.02)	0.82(0.001)	0.82(0.0008)	7.6(0.03)
Alt-BuresGAN	11.92(0.024)	11.12(0.04)	0.98(0.008)	1.15(0.007)	1.15(0.007)	10.10(0.03)

Table A.2: Inception Score for the best trained models on CIFAR-10, CIFAR-100 and STL-10, with a DCGAN architecture (higher is better).

	CIFAR-10	CIFAR-100	STL-10
GAN	5.92	6.33	6.13
WGAN-GP	2.54	2.56	/
UnrolledGAN	4.06	4.14	/
VEEGAN	3.51	3.85	/
GDPP	6.21	6.32	6.05
BuresGAN	6.69	6.67	7.94
Alt-BuresGAN	6.40	6.48	7.88

A.2.5 Influence of the number of convolutional layers for DCGAN architecture

Results for an architecture with 3 and 4 convolutional layers described in Table A.12 and Table A.6 are listed in the paper. An overview of similar experiments in other papers is given in Table A.9 hereafter. A more complete study is performed in this section. We provide here results with a DCGAN architecture for a simpler architecture with 2 convolutional layers for the discriminator (see, Table A.4 for the results and Table A.12 for the architecture).

Discussion. Interestingly, for most models, an improvement is observed in the quality of the images – KL divergence – and in terms of mode collapse – number of modes attained – as the size of the batch increases. For the same batch size, architecture and iterations, the image quality is improved by BuresGAN, which is robust with respect to batch size and architecture choice. The other methods show a higher variability over the different experiments. WGAN-GP has the best single run performance with a discriminator with 3 convolutional layers and has on average a superior performance when using a discriminator with 2 convolutional layers (see Table A.4) but sometimes fails to converge when increasing the number of discriminator layers by 1 along with increasing the batch size. MDGANv2, VEEGAN, GDPP and WGAN-GP often have an excellent single run performance. However, when increasing the number of discriminator layers, the training of these models has a tendency to collapse more often as indicated by the large standard deviation. Vanilla GAN is one of the best performing models in the variant with 3 layers. This indicates that, for certain datasets, careful architecture tuning can be more important than complicated training schemes. A lesson from Table A.4 is that BuresGAN's mode coverage does not vary much if the batch size increases, although the KL divergence seems to be slightly improved.

Ablation study. Results obtained with the Frobenius norm rather than the Bures metric are also included in the tables of this section.

Table A.3: Comparison for Frobenius norm. Experiments on the synthetic data sets. Average (std) over 10 runs. All the models are trained for 25k iterations.

	Grid (25 modes)		Ring (8 modes)	
	Nb modes	% in 3σ	Nb modes	% in 3σ
Frobenius^{2}GAN	25(0)	73(1)	8(0)	59(16)
BuresGAN	25(0)	82(1)	8(0)	82(4)
Alt-BuresGAN	**25**(0)	**84**(1)	**8**(0)	**84**(6)

Table A.4: KL-divergence between the generated distribution and true distribution (Quality, lower is better). The number of counted modes indicates the amount of mode collapse (higher is better). 25k iterations and average and std over 10 runs. Same architecture as in Table 2 in the paper with a discriminator with 2 convolutional layers. The architecture details are in Table A.12.

		Nb modes(↑)			KL div.(↓)		
	Batch size	64	128	256	64	128	256
2 conv. layers	GAN	970.5(5.8)	972.7(6.4)	979(3.5)	0.47(0.04)	0.44(0.02)	0.41(0.03)
	WGAN-GP	**996.7**(1.6)	**997.5**(0.9)	**998.1**(1.5)	**0.25**(0.01)	**0.22**(0.01)	**0.21**(0.05)
	MDGAN-v1	115.9(197)	260.9(267)	134.3(157)	5.5(1.4)	4.9(1.7)	5.8(0.9)
	MDGAN-v2	698.1(456)	898.4(299)	599.2(488)	2.2(3.0)	0.86(1.9)	2.8(3.2)
	UnrolledGAN	953.5(11)	971.4(4.8)	966.2(17.3)	0.71(0.06)	0.60(0.04)	0.58(0.10)
	VEEGAN	876.7(290)	688.5(443)	775.9(386)	0.92(1.6)	1.9(2.4)	1.54(2.2)
	GDPP	974.4(3.3)	978.2(7.6)	980.5(6.0)	0.45(0.02)	0.43(0.03)	0.41(0.03)
	PacGAN2	969.8(6.9)	971.1(3.6)	977.9(4.3)	0.54(0.04)	0.51(0.01)	0.48(0.02)
	Frobenius^{2}GAN	975.8(8.0)	981.0(4.1)	981.8(3.2)	0.34(0.03)	0.28(0.02)	0.25(0.01)
	BuresGAN	973.2(1.3)	979.9(4.0)	981.1(4.9)	0.36(0.02)	0.30(0.02)	0.25(0.01)
	Alt-BuresGAN	975.4(6.8)	978.2(5.4)	980.2(3.0)	0.37(0.02)	0.30(0.01)	0.28(0.01)

Table A.5: Ablation study on real image data sets for a convolutional architecture. Frobenius norm is also considered in the place of the Bures metric. Generation quality on CIFAR-10, CIFAR-100 and STL-10 with a DCGAN architecture. Average(std) over 10 runs. 100k iterations for each. For improving readability, SWD score was multiplied by 100.

	CIFAR-10			CIFAR-100			STL-10		
	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)
Frobenius^{2}GAN	5.40(0.16)	57.7(4.8)	**1.3**(0.3)	5.52(0.16)	60.1(4.4)	**1.6**(0.5)	6.26(0.15)	132.18(3.6)	**1.8**(0.4)
BuresGAN	**6.34**(0.17)	**43.7**(0.9)	2.1(0.6)	**6.5**(0.1)	**47.2**(1.2)	2.1(1.0)	**7.6**(0.3)	**109**(7)	2.3(0.3)
Alt-BuresGAN	6.23(0.07)	45.4(2.8)	1.7(0.9)	6.4(0.1)	49.4(3.4)	1.8(0.6)	7.5(0.3)	110(4)	2.8(0.4)

Table A.6: Ablation study for Stacked MNIST experiment for an architecture with 4 convolutional layers (see Table A.11). All the models are trained for 25k iterations with a batch size of 64, a learning rate of 2e−4 for Adam and a normal latent distribution. The evaluation is over 10k samples and we report an average (std) over 10 runs.

	Nb modes(↑)	KL div.(↓)
Frobenius^{2}GAN	984.7(6.4)	0.38(0.04)
BuresGAN	989.9(4.7)	0.38(0.06)
Alt-BuresGAN	990.0(4.9)	0.33(0.04)

A.3 Additional Figures

A.3.1 Training visualization at different steps for the artificial data sets

In Figure A.1 and Figure A.2, we provide snapshots of generated samples for the methods studied in the case of Ring and Grid. In both cases, BuresGAN seems to generate samples in such a way that the symmetry of the training data is preserved during training. This is also true for WGAN and MDGAN, while some other GANs seem to recover the symmetry of the problem at the end of the training phase.

A.3.2 Generated images of the trained GANs

Next, in Figure A.3, Figure A.4, Figure A.5 and Figure A.6, we list samples of generated images for the models trained with a DCGAN architecture for Stacked MNIST, CIFAR-10, CIFAR-100 and STL-10 respectively.

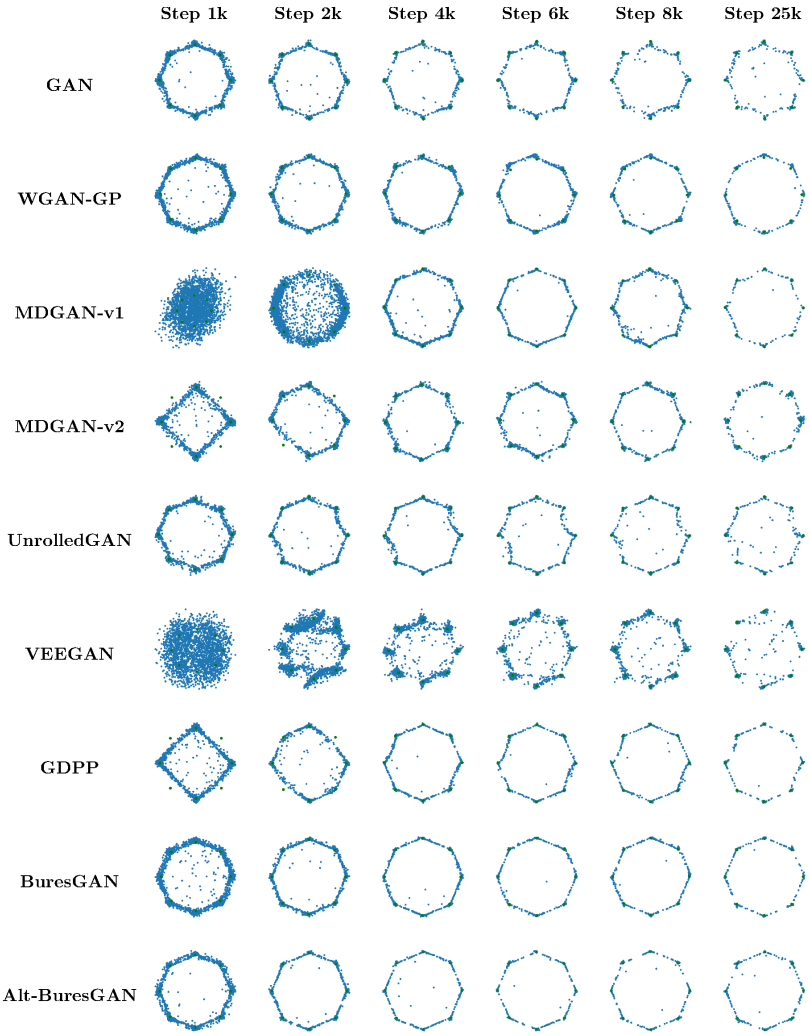


Figure A.1: The progress of different GANs on the synthetic ring example. Each column show 3000 samples from the training generator in blue with 3000 samples from the true distribution in green.

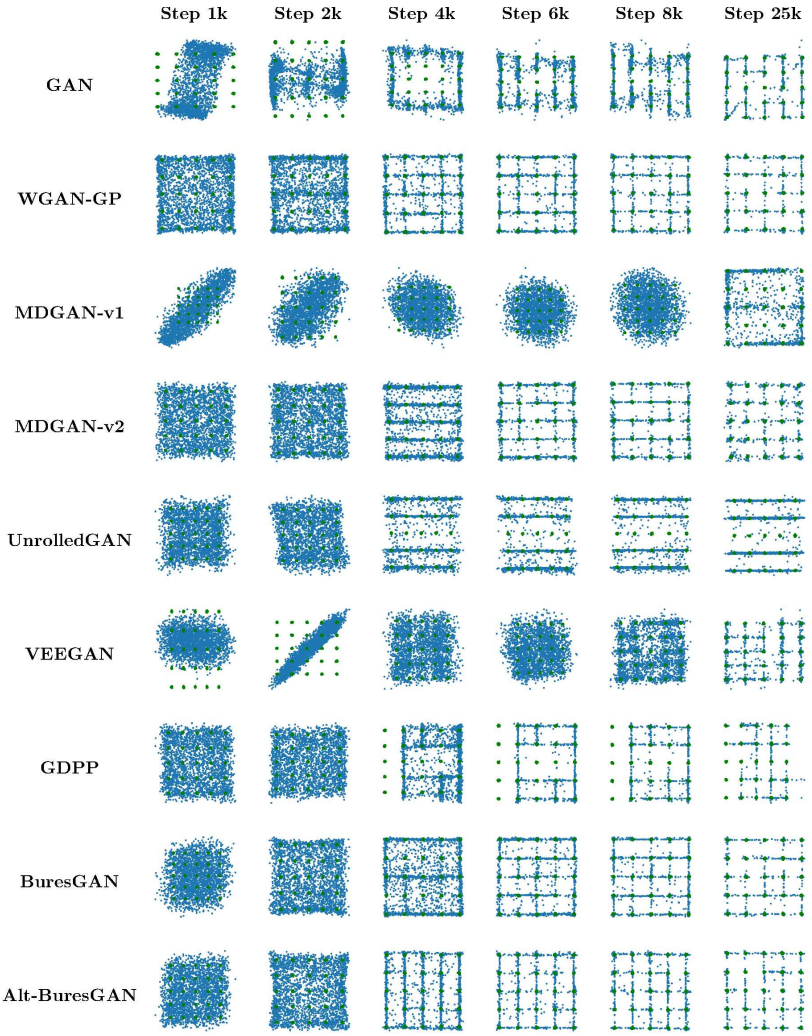


Figure A.2: The progress of different GANs on the synthetic grid example. Each column show 3000 samples from the training generator in blue with 3000 samples from the true distribution in green.

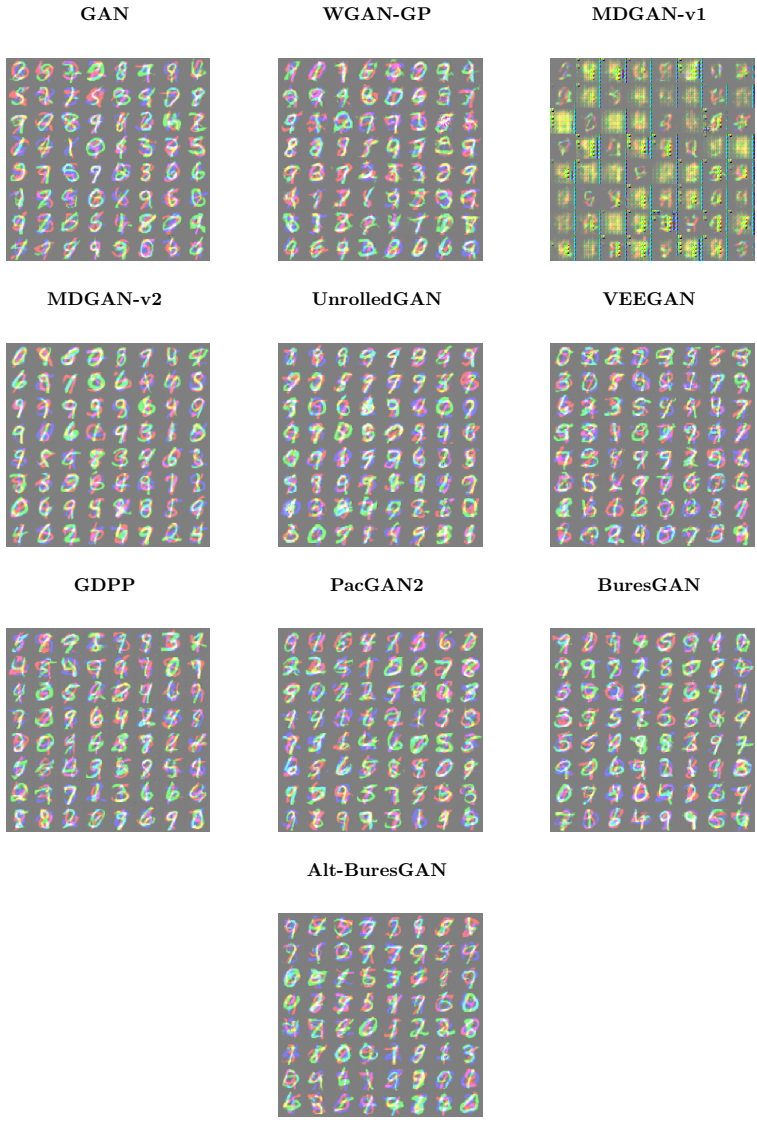


Figure A.3: Generated images for the Stacked MNIST dataset. Each model is trained with 3 layers and mini-batch size 256. Each square shows 64 samples from the trained generator.



Figure A.4: Generated images for CIFAR-10 using a DCGAN architecture. Each square shows 64 samples from the trained generator.



Figure A.5: Generated images for CIFAR-100 using a DCGAN architecture. Each square shows 64 samples from the trained generator.

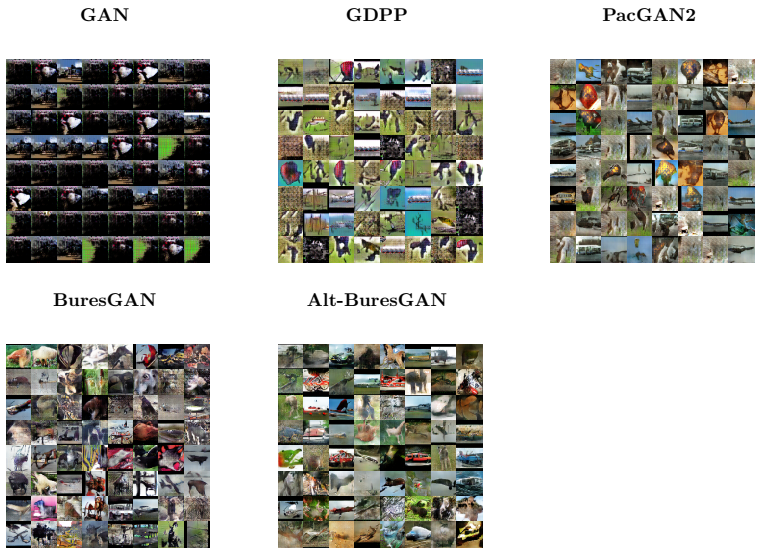


Figure A.6: Generated images for STL-10 using a DCGAN architecture. Each square shows 64 samples from the trained generator.

A.4 Architectures

A.4.1 Synthetic Architectures

Following the recommendation in the original work of [195], the same fully-connected architecture is used for the VEEGAN reconstructor in all experiments.

Table A.7: The generator and discriminator architectures for the synthetic examples.

Layer	Output	Activation
Input	256	-
Dense	128	tanh
Dense	128	tanh
Dense	2	-

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	128	tanh
Dense	1	-

Table A.8: Respectively the MDGAN encoder model and VEEGAN stochastic inverse generator architectures for the synthetic examples. The output of the VEEGAN models are samples drawn from a normal distribution with scale 1 and where the location is learned.

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	256	-

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	128	tanh
Dense	256	tanh
Normal	256	-

A.4.2 Stacked MNIST Architectures

For clarity, we report in Table A.9 the settings of different papers for the Stacked MNIST experiment. This helps to clarify the difference between our simulations and other papers' simulations. On the one hand, in Table A.11, the architecture of the experiment reported in the paper is described. On the other hand, the architecture of the supplementary experiments on Stacked MNIST is given

in Table A.12. Details of the architecture of MDGAN are also reported in Table A.10.

Table A.9: The hyperparameters and architectures of the stacked MNIST experiment performed by the different methods are given here. This represents our best efforts at providing a comparison between the different parameters used. The asterisks ** indicate that the parameters where obtained by the Github repository. Notice that GDPP paper also used 30000 iterations for training DCGAN and unrolled GAN (indicated by *). BuresGAN's column refers to the settings of experiment of Table 2 in the paper and Table A.6 respectively, for which the different values are separated by the symbol &.

Parameters	BuresGAN	PacGAN	VEEGAN	GDPP
learning rates	1×10^{-3} & 2×10^{-4}	2×10^{-4}	2×10^{-4}	1×10^{-4} **
learning rate decay	no	no	no	no**
Adam β_1	0.5	0.5	0.5	0.5
Adam β_2	0.999	0.999	0.999	0.9
iterations	25000	20000	?	15000*
disc. conv. layers	3 & 4	4	4	3
gen. conv. layers	3 & 4	4	4	3
batch size	64, 128, 256 & 64	64	64	64
evaluation samples	10000	26000	26000	26000
ℓ (i.e., z dimension)	100	100	100	128
z distribution	$\mathcal{N}(0, \mathbb{I}_\ell)$	unif$[-1,1]^\ell$	unif$[-1,1]^\ell$ **	unif$[-1,1]^\ell$

Table A.10: The MDGAN encoder model architecture for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	28, 28, 3	-	-
Conv	14, 14, 3	ReLU	Yes
Conv	7, 7, 64	ReLU	Yes
Conv	7, 7, 128	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

Table A.11: The generator and discriminator architectures for the Stacked MNIST experiments with 4 layers for both generator and discrimintor. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	100	-	-
Dense	2048	ReLU	Yes
Reshape	2, 2, 512	-	-
Conv'	4, 4, 256	ReLU	Yes
Conv'	7, 7, 128	ReLU	Yes
Conv'	14, 14, 64	ReLU	Yes
Conv'	28, 28, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	28, 28, 3	-	-
Conv	14, 14, 64	Leaky ReLU	No
Conv	7, 7, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Conv	2, 2, 512	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table A.12: The generator and discriminator architectures for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not. For the experiments with 2 convolution layers in Table A.4, the final convolution layer is removed in the discriminator.

Layer	Output	Activation	BN
Input	100	-	-
Dense	12544	ReLU	Yes
Reshape	7, 7, 256	-	-
Conv'	7, 7, 128	ReLU	Yes
Conv'	14, 14, 64	ReLU	Yes
Conv'	28, 28, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	28, 28, 3	-	-
Conv	14, 14, 64	Leaky ReLU	No
Conv	7, 7, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table A.13: The CNN architecture of the classifier used during the evaluation of the stackedMNIST experiments. Dropout with a rate of 0.5 is used before the final dense layer.

Layer	Output	Activation
Input	28, 28,1	-
Conv	24, 24, 32	ReLU
MaxPool	12, 12, 32	-
Conv	8, 8, 64	ReLU
MaxPool	4, 4, 64	-
Flatten	-	-
Dense	1024	ReLU
Dense	10	-

A.4.3 CIFAR-10 and 100 DCGAN Architectures

Convolutional architectures for CIFAR data set are reported in Table A.14 and Table A.15 below.

Table A.14: The generator and discriminator architectures for the CIFAR-10 and CIFAR-100 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	100	-	-
Dense	16384	ReLU	Yes
Reshape	8, 8, 256	-	-
Conv'	8, 8, 128	ReLU	Yes
Conv'	16, 16, 64	ReLU	Yes
Conv'	32, 32, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	32, 32, 3	-	-
Conv	16, 16, 64	Leaky ReLU	No
Conv	8, 8, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table A.15: The MDGAN encoder model architecture for the CIFAR-10 and CIFAR-100 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	32, 32, 3	-	-
Conv	16, 16, 3	ReLU	Yes
Conv	8, 8, 64	ReLU	Yes
Conv	8, 8, 128	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

A.4.4 STL-10 DCGAN Architectures

In Table A.16, the convolutional architecture is described for the $96 \times 96 \times 3$ STL-10 data set, while MDGAN details are reported in Table A.17.

Table A.16: The generator and discriminator architectures for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	100	-	-
Dense	36864	ReLU	Yes
Reshape	12, 12, 256	-	-
Conv'	12, 12, 256	ReLU	Yes
Conv'	24, 24, 128	ReLU	Yes
Conv'	48, 48, 64	ReLU	Yes
Conv'	96, 96, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	96, 96, 3	-	-
Conv	48, 48, 64	Leaky ReLU	No
Conv	24, 24, 128	Leaky ReLU	Yes
Conv	12, 12, 256	Leaky ReLU	Yes
Conv	6, 6, 512	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table A.17: The MDGAN encoder model architecture for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	96, 96, 3	-	-
Conv	48, 48, 3	ReLU	Yes
Conv	24, 24, 64	ReLU	Yes
Conv	12, 12, 128	ReLU	Yes
Conv	12, 12, 256	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

A.4.5 ResNet Architectures

For CIFAR-10, we used the ResNet architecture from the appendix of [69] with minor changes as given in Table A.18. We used an initial learning rate of 5e−4 for CIFAR-10 and STL-10. For both datasets, the models are run for 200k iterations. For STL-10, we used a similar architecture that is given in Table A.19.

Table A.18: The generator (top) and discriminator (bottom) ResNet architectures for the CIFAR-10 experiments.

Layer	Kernel Size	Resample	Output Shape
Input	-	-	128
Dense	-	-	$200 \cdot 4 \cdot 4$
Reshape	-	-	$200 \times 4 \times 4$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 8 \times 8$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 16 \times 16$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 32 \times 32$
Conv, tanh	3×3	-	$3 \times 32 \times 32$

Layer	Kernel Size	Resample	Output Shape
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 16 \times 16$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 8 \times 8$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 4 \times 4$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 4 \times 4$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 4 \times 4$
ReLu, meanpool	-	-	200
Dense	-	-	1

Table A.19: The generator (top) and discriminator (bottom) ResNet architectures for the STL-10 experiments. For the experiment with full-sized 96x96 images, an extra upsampling block was added to the generator.

Layer	Kernel Size	Resample	Output Shape
Input	-	-	128
Dense	-	-	$200 \cdot 6 \cdot 6$
Reshape	-	-	$200 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 12 \times 12$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 24 \times 24$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 48 \times 48$
Conv, tanh	3×3	-	$3 \times 48 \times 48$

Layer	Kernel Size	Resample	Output Shape
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 24 \times 24$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 12 \times 12$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 6 \times 6$
ReLu, meanpool	-	-	200
Dense	-	-	1

Appendix B

Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks

B.1 Organization

In Section B.2, the remaining results from the UNBALANCED MNIST experiment are given. In Section B.3, the effect of the ridge parameter γ and dimensionality reduction size k is discussed. Then, in Section B.4, the training times of our simulations are reported. Section B.5 provides further information about the distribution of generated samples in the artificial datasets. Finally, Section B.6 describes the experimental setting, the proposed algorithm, and architectures used in the paper.

B.2 Extra table unbalanced MNIST dataset

For the UNBALANCED MNIST experiment, only the number of samples in the minority modes are given in the main part. The number of samples for the remaining modes is given in Table B.1.

Table B.1: Experiments on the UNBALANCED MNIST dataset. Two variants of RLS BuresGAN are trained. First, RLSs are calculated with an explicit feature map obtained from a pre-trained classifier (Class.). The second version uses the next-to-last layer of the discriminator (Discr.) to compute the RLSs. The RLS MwuGAN is initialized using the RLSs with the explicit feature maps obtained from a pre-trained classifier.

	Mode 6	Mode 7	Mode 8	Mode 9	Mode 10
GAN	1831(135)	1835(32)	2009(69)	1747(46)	1915(143)
PacGAN2	1767(116)	1864(16)	1983(105)	1847(25)	1900(110)
BuresGAN	1762(96)	1870(38)	1979(110)	1782(29)	1916(68)
IwGAN	1833(74)	1837(97)	1902(79)	1783(64)	1928(74)
IwMmdGAN	67(45)	0(0)	4(3)	8741(430)	22(10)
MwuGAN (15)	1638(137)	1986(114)	1807(96)	2026(22)	1801(123)
RLS MwuGAN (15) (ours)	1616(17)	1513(144)	2018(78)	1576(30)	1811(34)
RLS BuresGAN Class. (ours)	1311(178)	1137(116)	1743(79)	1069(67)	1396(100)
RLS BuresGAN Discr. (ours)	2012(97)	1839(53)	1608(155)	1841(55)	1545(115)

B.3 Effect of the regularization parameter in RLS

B.3.1 Synthetic data

We display in Table B.2 an ablation study for different ridge regularization parameters $\gamma > 0$ on the synthetic RING and GRID. A lesson from Table B.2 is that the performance on the synthetic datasets does not vary much if different values of γ are chosen in the case of the discriminator feature map. Notice that, for $\gamma = 10^{-3}$ and $\gamma = 10^{-4}$, there is a small improvement in sample quality for RING and in terms of mode coverage for GRID with the Gaussian kernel feature map.

Impact of the number of generators in MwuGAN. RLSs can also be used as initial starting weights in MwuGAN. The RLSs are constructed with a Gaussian kernel with bandwidth $\sigma = 0.15$ and regularization $\gamma = 10^{-3}$. Both methods contain a mixture of 15 GANs. This approach is compared with the classical initialization with uniform weights. In Figure B.1, a clear improvement over a uniform initialization is visible when RLSs are used as initial weights.

Table B.2: Ablation study over the parameter γ on the synthetic datasets for RLS BuresGAN with a discriminator feature map (Discr.) and Gaussian feature map (Gauss.) with bandwidth $\sigma = 0.15$. No significant difference can be seen from the reported performance as the regularization parameter varies.

Feature map	γ	Ring with 8 modes		Grid with 25 modes	
		Nb modes	% in 3σ	Nb modes	% in 3σ
Discr.	10^{-2}	8(0)	0.92(0.02)	24.0(1.3)	0.79(0.03)
	10^{-3}	8(0)	0.90(0.02)	24.4(0.9)	0.78(0.06)
	10^{-4}	8(0)	0.90(0.01)	24.5(0.8)	0.77(0.07)
Gauss.	10^{-2}	8(0)	0.83(0.10)	19.9(1.9)	0.76(0.09)
	10^{-3}	8(0)	0.90(0.02)	24.0(1.5)	0.76(0.11)
	10^{-4}	8(0)	0.87(0.04)	24.8(0.6)	0.78(0.03)

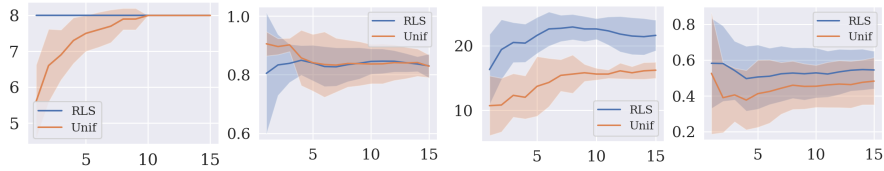


Figure B.1: RING: number of modes (first) and sample quality (second). GRID: number of modes (third) and sample quality (fourth). The x-axis refers to the number of generators of MwuGAN. Sample quality is assessed by counting the number of modes within 3σ of each mode. The blue (resp. red) curve indicates the results obtained for RLS (resp. unif.) sampling with MwuGAN.

B.3.2 Unbalanced MNIST

We describe here an ablation study for different ridge regularization parameters $\gamma > 0$ and size of dimension reduction k on the UNBALANCED 012-MNIST and UNBALANCED MNIST dataset. Moreover, the use of sketching the fixed explicit feature map is analyzed. From these results is concluded that the non-linear UMAP procedure leads to better mode coverage. The results are given in Tables B.3 and B.4.

Table B.3: Ablation study over the regularization parameter γ and dimension k on the UNBALANCED 012-MNIST dataset for RLS BuresGAN with a discriminator feature map (Discr.) with sketching and an explicit feature map obtained from the next-to-last layer of a pre-trained classifier (Class.) with a UMAP dimensionality reduction and sketched dimensionality reduction. Minority modes are highlighted in black in the first row. The best performance is achieved using the Discr. sketched and Class. UMAP. feature maps with regularization 10^{-4} and dimension $k = 25$.

Feature map	Dim.	γ	Mode 1	Mode 2	**Mode 3**	KL
Discr. Sketched	10	10^{-2}	5561(381)	2756(300)	1409(345)	0.16(0.05)
		10^{-3}	5568(298)	2769(178)	1386(198)	0.15(0.03)
		10^{-4}	5655(279)	2812(135)	1218(253)	0.17(0.04)
	25	10^{-2}	5814(379)	2405(225)	1501(247)	0.17(0.04)
		10^{-3}	5715(312)	2477(139)	1415(310)	0.17(0.04)
		10^{-4}	5748(172)	2416(268)	1566(293)	0.16(0.02)
Class. UMAP	10	10^{-2}	3711(173)	5361(103)	751(161)	0.18(0.03)
		10^{-3}	3813(195)	5425(208)	644(67)	0.19(0.01)
		10^{-4}	3713(221)	5431(226)	712(152)	0.18(0.03)
	25	10^{-2}	4221(104)	4958(137)	672(109)	0.18(0.02)
		10^{-3}	4179(111)	4876(148)	802(105)	0.16(0.01)
		10^{-4}	3414(161)	4862(134)	1461(183)	0.08(0.01)
Class. Sketched	10	10^{-2}	5865(122)	3389(79)	605(111)	0.25(0.03)
		10^{-3}	5715(241)	3456(216)	683(161)	0.31(0.01)
		10^{-4}	5952(309)	3403(206)	513(135)	0.27(0.04)
	25	10^{-2}	5998(193)	3130(105)	672(119)	0.25(0.03)
		10^{-3}	5957(100)	3127(136)	775(95)	0.23(0.01)
		10^{-4}	6009(157)	3138(142)	698(121)	0.32(0.01)

Table B.4: Ablation study over the regularization parameter γ and dimension k on the UNBALANCED MNIST dataset for RLS BuresGAN with a discriminator feature map (Discr.) with sketching and an explicit feature map obtained from the next-to-last layer of a pre-trained classifier (Class.) with a UMAP dimensionality reduction and sketched dimensionality reduction. Minority modes are highlighted in black in the first row. The best performance is achieved using the Class. UMAP. feature map.

Feature map	Dim.	γ	**Mode 1**	**Mode 2**	**Mode 3**	**Mode 4**	**Mode 5**	Mode 6	Mode 7	Mode 8	Mode 9	Mode 10	KL
Discr. Sketched	10	10^{-2}	254(77)	170(30)	222(40)	227(46)	212(62)	2061(94)	1871(134)	1598(71)	1853(83)	1532(96)	0.38(0.02)
		10^{-3}	214(38)	172(123)	239(35)	223(99)	230(54)	2081(91)	1832(32)	1611(100)	1898(44)	1501(123)	0.39(0.02)
		10^{-4}	235(62)	183(141)	264(44)	255(109)	219(54)	2012(97)	1839(53)	1608(155)	1841(55)	1545(115)	0.37(0.02)
	25	10^{-2}	236(56)	191(145)	286(43)	231(72)	219(60)	2066(113)	1765(48)	1505(81)	1958(37)	1543(56)	0.37(0.02)
		10^{-3}	230(30)	172(129)	267(31)	248(89)	274(68)	2046(100)	1780(42)	1546(122)	1899(73)	1538(135)	0.37(0.02)
		10^{-4}	224(49)	147(179)	255(27)	223(52)	197(53)	2032(116)	1790(49)	1554(119)	1962(55)	1617(143)	0.40(0.02)
Class. UMAP	10	10^{-2}	885(79)	662(143)	359(229)	814(97)	578(66)	1248(152)	1137(152)	1847(154)	1095(98)	1374(112)	0.10(0.01)
		10^{-3}	892(137)	674(90)	370(227)	803(108)	605(75)	1187(189)	1123(103)	1784(111)	1110(95)	1452(189)	0.09(0.02)
		10^{-4}	875(112)	663(122)	360(198)	831(59)	615(82)	1311(178)	1137(116)	1743(79)	1069(67)	1396(100)	0.09(0.01)
	25	10^{-2}	556(60)	525(82)	352(59)	542(71)	473(56)	1471(99)	1449(87)	1915(101)	1327(66)	1392(80)	0.16(0.01)
		10^{-3}	495(58)	473(114)	385(50)	546(83)	463(50)	1450(120)	1452(69)	1964(110)	1290(75)	1483(81)	0.17(0.02)
		10^{-4}	499(73)	443(103)	358(48)	513(104)	450(45)	1491(148)	1468(51)	1957(119)	1329(43)	1493(145)	0.18(0.01)
Class. Sketched	10	10^{-2}	218(72)	188(171)	165(31)	261(114)	246(41)	2019(166)	1726(48)	1859(204)	1683(63)	1637(178)	0.38(0.02)
		10^{-3}	189(45)	194(238)	184(88)	245(232)	221(44)	2147(176)	1712(38)	1824(182)	1666(61)	1618(177)	0.4(0.03)
		10^{-4}	239(45)	127(200)	211(27)	232(133)	214(65)	2127(113)	1783(48)	1804(122)	1642(29)	1622(142)	0.48(0.02)
	25	10^{-2}	238(30)	192(205)	189(35)	308(84)	254(39)	2011(105)	1699(74)	1710(127)	1785(54)	1615(143)	0.36(0.02)
		10^{-3}	262(45)	204(116)	251(45)	258(83)	266(62)	2041(152)	1621(40)	1774(119)	1751(41)	1573(134)	0.35(0.02)
		10^{-4}	246(50)	220(131)	199(41)	312(70)	306(41)	2033(127)	1689(38)	1712(115)	1724(67)	1560(100)	0.34(0.02)

B.4 Timings

Training times with a single NVIDIA Tesla P100-SXM2-16GB @1.3 GHz GPU. Temporary evaluations and plotting during training are included in the timings, so only relative comparisons are instructive.

Table B.5: Total training time in seconds, averaged over 10 runs.

	Ring	Grid	012-MNIST	MNIST
GAN	45(1)	48(1)	1063(4)	1074(3)
PacGAN2	65(1)	71(1)	1205(4)	1211(8)
BuresGAN	100(1)	105(2)	1357(3)	1373(3)
IwGAN	65(1)	65(1)	1066(5)	1080(3.3)
IwMmdGAN	45(1)	56(1)	519(4)	5126(1)
MwuGAN (15)	1195(9)	1225(8)	20730(314)	36074(292)
RLS MwuGAN (15)	1198(9)	1217(9)	20823(233)	35400(566)
RLS BuresGAN Gauss.	130(3)	136(2)	/	/
RLS BuresGAN Class.	/	/	1615(11)	2494(8)
RLS BuresGAN Discr.	176(4)	187(1)	2006(3)	2004(2)

B.5 Distribution of generated samples for the artificial datasets

The number of samples per mode for all GANs in the Ring experiments are given in Figure B.2. The number of samples per mode for all GANs in the Grid experiments are given in Figure B.3.

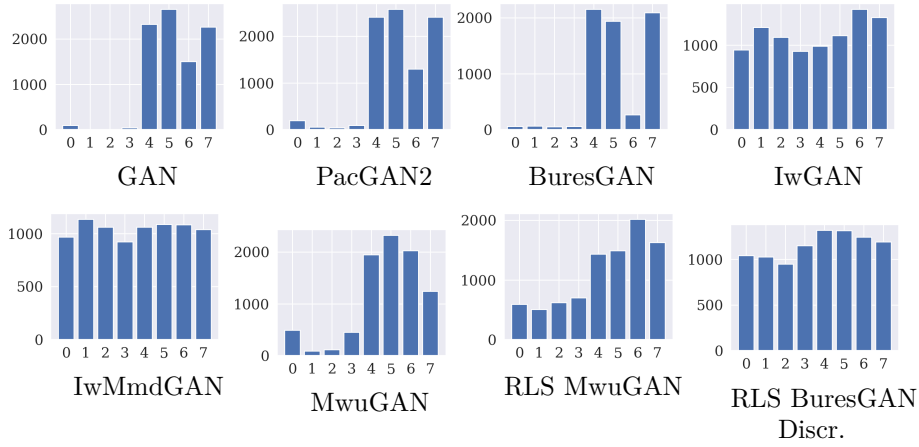


Figure B.2: Samples per mode for a single GAN trained on the Ring dataset. Only IwMmdGAN, MwuGAN (15), RLS MwuGAN (15), and RLS BuresGAN with a discriminator feature map (Discr.) are capable of covering all the modes.

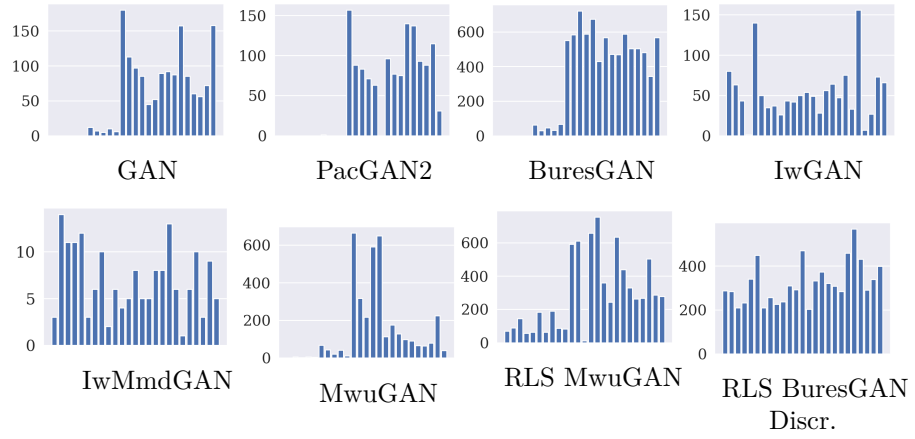


Figure B.3: Samples per mode for a single GAN trained on the Grid dataset. Only the proposed RLS BuresGAN with a discriminator feature map (Discr.) is capable of covering all the modes.

B.6 Architectures, algorithms, and settings

Settings experimental section. Unless specified otherwise, the models are trained for 30k iterations with a batch size of 64, by using the Adam optimizer

with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and learning rate 10^{-3} for both the generator and discriminator. The dimensionality of the generator latent space ℓ is equal to 100. The results of the experimental section always refer to the performance achieved at the end of the training. All the images are scaled in between -1 and 1 before running the algorithms. The hyperparameters of the competing methods are chosen as suggested in the authors' reference implementations. For MwuGAN, we take $\delta = 0.25$ and run for an increasing number of generators (displayed in brackets). In all the simulations, PacGAN always uses 2 as packing number. Note that IwMmdGAN and IwGAN require full knowledge of the Radon-Nikodym derivative M, where we want to generate data from a target distribution p but only have access to representative samples from a modified distribution Mp. The method is thus not an unsupervised method and tackles a slightly different problem. However, the method was included for completeness.

Algorithm B.1: RLS BuresGAN with a fixed feature map.

Input: $\{x_i\}_{i=1}^n$, regularization γ, dimension reduction size k, discriminator D, generator G, feature map $\varphi(\cdot) \in \mathbb{R}^m$ - explicit *or* implicit via kernels $k(\cdot, \cdot)$
Calculate ridge leverage scores.
if $\varphi(\cdot)$ = *Implicit* **then**
  Calculate the RLSs for all points: $\ell_i = \left(K(K + n\gamma\mathbb{I})^{-1}\right)_{ii}$.
else if $\varphi(\cdot)$ = *Explicit* **then**
  if *dimension reduction* **then**
    $\varphi(x_i) \leftarrow$ UMAP of size k calculated on the full projected dataset $\{\varphi(x_i)\}_{i=1}^n$.
  else
    $k \leftarrow m$.
  if $n > k$ **then**
    Calculate the RLSs for all points: $\ell_i = \varphi(x_i)^\top (C + n\gamma\mathbb{I})^{-1}\varphi(x_i)$.
  else
    Calculate the RLSs for all points: $\ell_i = \left(K(K + n\gamma\mathbb{I})^{-1}\right)_{ii}$.
Train BuresGAN.
while *not converged* **do**
  Sample a real and fake batch with probability $x_i \sim \ell_i$.
  Update the generator G by minimizing $V_G + \mathcal{B}(\hat{C}_r, \hat{C}_g)^2$
  Update the discriminator D by maximizing $-V_D$

Algorithm B.2: RLS BuresGAN with a discriminator-based explicit feature map.

Input: $\{x_i\}_{i=1}^n$, regularization γ, dimension reduction size k, mini-batch size s, discriminator D with discriminator feature map $\varphi_D(\cdot) \in \mathbb{R}^m$, generator G.
Train BuresGAN with adaptive RLSs.
while *not converged* **do**
  Sample a mini-batch $\hat{x}$ uniform at random of size $b = 20s$.
  if *dimension reduction* **then**
    $\varphi(\hat{x}_i) \leftarrow S^\top \varphi_D(\hat{x}_i) \in \mathbb{R}^k$, with sketching matrix $S = A/\sqrt{k} \in \mathbb{R}^{m \times k}$.
  else
    $\varphi(\hat{x}_i) \leftarrow \varphi_D(\hat{x}_i)$, $k \leftarrow m$.
  if $b > k$ **then**
    Calculate the RLSs for all points: $\ell_i = \varphi(x_i)^\top (C + n\gamma \mathbb{I})^{-1} \varphi(x_i)$.
  else
    Calculate the RLSs for all points: $\ell_i = \left(K(K + n\gamma \mathbb{I})^{-1}\right)_{ii}$.
  Sample a real and fake batch of size s with probability $\hat{x}_i \sim \ell_i$.
  Update the generator G by minimizing $V_G + \mathcal{B}(\hat{C}_r, \hat{C}_g)^2$
  Update the discriminator D by maximizing $-V_D$

Table B.6: The generator (left) and discriminator (right) architectures for the synthetic examples.

Layer	Output	Activation
Input	25	-
Dense	128	tanh
Dense	128	tanh
Dense	2	-

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	128	tanh
Dense	1	-

Table B.7: The generator (left) and discriminator (right) architectures for the UNBALANCED 012-MNIST and the UNBALANCED MNIST experiments. BN indicates if batch normalization is used.

Layer	Output	Activation	BN
Input	100	-	-
Dense	12544	ReLU	Yes
Reshape	7, 7, 256	-	-
Conv'	7, 7, 128	ReLU	Yes
Conv'	14, 14, 64	ReLU	Yes
Conv'	28, 28, 1	ReLU	Yes

Layer	Output	Activation	BN
Input	28, 28, 1	-	-
Conv	14, 14, 64	Leaky ReLU	No
Conv	7, 7, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Conv	2, 2, 512	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table B.8: The CNN architecture of the classifier used during the evaluation of the MNIST experiments. Dropout with a rate of 0.5 is used before the final dense layer.

Layer	Output	Activation
Input	28, 28,1	-
Conv	24, 24, 32	ReLU
MaxPool	12, 12, 32	-
Conv	8, 8, 64	ReLU
MaxPool	4, 4, 64	-
Flatten	-	-
Dense	1024	ReLU
Dense	10	-

Appendix C

Multi-view Kernel PCA for Time series Forecasting

C.1 Note on kernel centering

Here we derive the expression for centering the kernel matrix and kernel vector using matrix products. This is computationally more efficient than calculating the mean and subtracting it from each feature vector.

Centering Kernel Matrix Let $\Phi = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)]$ and the sample mean be $\boldsymbol{\mu}_\phi = \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i) = \frac{1}{n}\Phi\mathbb{1}_n$, where $\mathbb{1}_n = [1, \ldots, 1]^\top$. Let $\tilde{\phi}(\boldsymbol{x}_i) = \phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\phi$ denote the centered feature vector. Then

$$\tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_i) = k(\boldsymbol{x}_i, \boldsymbol{x}_i) - \langle\phi(\boldsymbol{x}_i), \boldsymbol{\mu}_\phi\rangle_{\mathscr{H}} - \langle\boldsymbol{\mu}_\phi, \phi(\boldsymbol{x}_i)\rangle_{\mathscr{H}} - \langle\boldsymbol{\mu}_\phi, \boldsymbol{\mu}_\phi\rangle_{\mathscr{H}}$$

$$\tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_i) = k(\boldsymbol{x}_i, \boldsymbol{x}_i) - \langle\phi(\boldsymbol{x}_i), \Phi\mathbb{1}_n\rangle_{\mathscr{H}}\frac{1}{n} - \frac{1}{n}\langle\Phi\mathbb{1}_n, \phi(\boldsymbol{x}_i)\rangle_{\mathscr{H}} + \frac{1}{n}\langle\Phi\mathbb{1}_n, \Phi\mathbb{1}_n\rangle_{\mathscr{H}}\frac{1}{n}$$

Composing the above in the matrix form $\forall i$, gives the following

$$\tilde{\boldsymbol{K}} = \boldsymbol{K} - \boldsymbol{K}\mathbb{1}\mathbb{1}^\top\frac{1}{n} - \frac{1}{n}\mathbb{1}\mathbb{1}^\top\boldsymbol{K} + \frac{1}{n^2}\mathbb{1}\mathbb{1}^\top\boldsymbol{K}\mathbb{1}\mathbb{1}^\top$$

$$\tilde{\boldsymbol{K}} = \big(\mathbb{I} - \mathbb{1}\mathbb{1}^\top\frac{1}{n}\big)\boldsymbol{K}\big(\mathbb{I} - \mathbb{1}\mathbb{1}^\top\frac{1}{n}\big) \tag{C.1}$$

Centering Kernel Vector Using the same sample statistic $\boldsymbol{\mu}_\phi$, a kernel vector representing the similarities $\boldsymbol{k}_{\boldsymbol{x}_i}(\boldsymbol{x}^\star) = [k(\boldsymbol{x}_1, \boldsymbol{x}^\star), \ldots, k(\boldsymbol{x}_n, \boldsymbol{x}^\star)]^\top$ can be centered as follows :

$$\tilde{\boldsymbol{k}}_{\boldsymbol{x}_i}(\boldsymbol{x}^\star) = \boldsymbol{k}_{\boldsymbol{x}_i}(\boldsymbol{x}^\star) - \boldsymbol{K}\mathbb{1}\frac{1}{n} - \frac{1}{n}\mathbb{1}\mathbb{1}^\top \boldsymbol{k}_{\boldsymbol{x}_i}(\boldsymbol{x}^\star) + \frac{1}{n^2}\mathbb{1}\mathbb{1}^\top \boldsymbol{K}\mathbb{1} \tag{C.2}$$

C.2 Datasets and Hyperparameters

We refer to Table C.1 for more details on model architectures, datasets and hyperparameters used in this paper. The PyTorch library (double precision) in Python was used. See Algorithm 3.1 for training the MV-RKM model.

Table C.1: Datasets used for the experiments. N is the number of training samples, d the input dimension, s the subspace dimension and m the minibatch size.

Dataset	N_{train}	N_{test}	d
SantaFe	1000	100	1
Lorenz	2801	1200	3
Chickenpox	418	104	20
Energy	4000	1000	28
Gas Turbine	5929	1482	11

Lorenz attractor is defined by the following dynamical system:

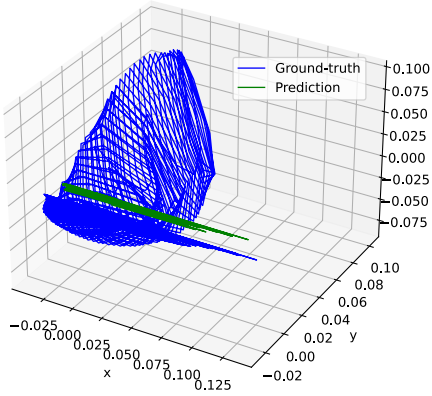
$$\begin{cases} \dot{x}(t) = -ax + ay \\ \dot{y}(t) = -xz + rx - y \\ \dot{z}(t) = xy - bz. \end{cases}$$

For our simulations, we set $a = 10$, $r = 28$ and $b = 2.667$ with initial condition $x(0) = 1.0$, $y(0) = -1.0$ and $z(0) = 1.05$ and the above equations are solved using the first-order finite difference scheme.

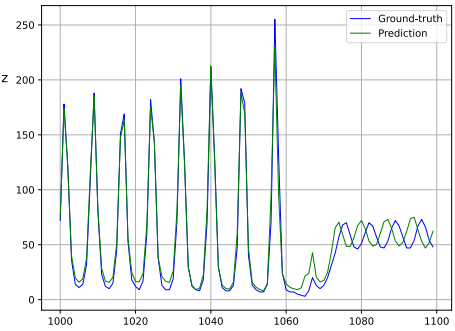
C.3 Least-Squares Support Vector Machine

We consider the non-linear auto-regressive setting for LS-SVM regression model: $\boldsymbol{y} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i k_x(\boldsymbol{x}_i, \boldsymbol{x}) + \boldsymbol{b}$, where the parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are found by solving the

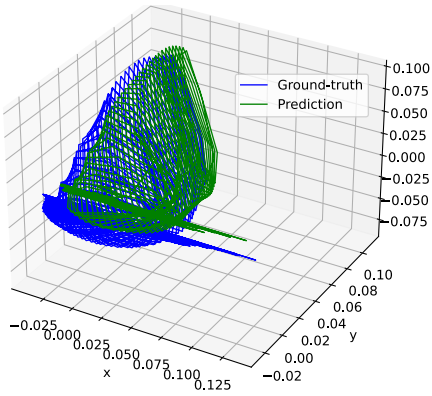
following linear system: $\left[\begin{array}{c|c} 0 & \mathbf{1}_n^\top \\ \hline \mathbf{1}_n & \boldsymbol{K}(\mathcal{X}) + \gamma^{-1}\mathbb{I} \end{array}\right] \left[\begin{array}{c} \boldsymbol{b} \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \boldsymbol{y} \end{array}\right]$. For details on the derivation from a constrained optimization problem, we refer to [199, p. 98].



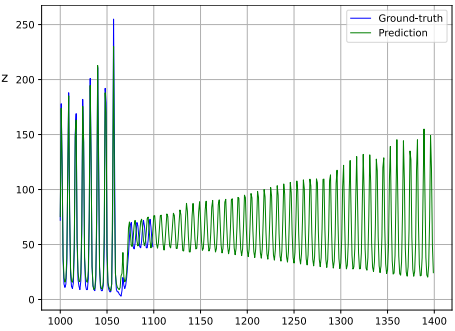
(a) Latent space



(b) Predictions



(c) Latent space



(d) Predictions

Figure C.1: MV-RKM (RBF $k_x(\sigma = 2.1856)$, linear k_y, $s = 144$, lag 70) on the SantaFe chaotic laser dataset showing short-term and long-term predictions in the first and second row respectively.

Appendix D

Explainable Unsupervised Anomaly Detection for Healthcare Insurance Data

D.1 Shapley Values

This section provided a brief introduction to Shapley values and how they are used to explain the predictions of the anomaly detection algorithms used in the workflow discussed.

The Shapley value [189] originates from cooperative game theory where its goal is to fairly distribute the reward among players when they form a coalition. For a fair distribution, this reward must coincide with the respective contribution to the game for each player.

Consider a cooperative game with M players and corresponding value function v that indicates the contribution (or total payoff) of forming a coalition $\mathcal{S} \subseteq \mathcal{M} = \{1, \ldots, M\}$. The *marginal* contribution of a player i to a coalition $\mathcal{S} \subseteq \mathcal{M} \backslash \{i\}$ is then defined as $v(\mathcal{S} \cup \{i\}) - v(\mathcal{S})$. This is the additional value that a player brings to an existing coalition if it would join that coalition. Considering this, the Shapley value ϕ_i for a player i is defined as the weighted mean of the player's marginal contributions over all possible coalitions $\mathcal{S} \subseteq \mathcal{M} \backslash \{i\}$:

$$\phi_i = \sum_{\mathcal{S} \subseteq \mathcal{M} \setminus \{i\}} \frac{|\mathcal{S}|!(M - |\mathcal{S}| - 1)!}{M!} (v(\mathcal{S} \cup \{i\}) - v(\mathcal{S})), \quad i = 1, \ldots, M \tag{D.1}$$

With $|\mathcal{S}|$ indicating the size of the subset $\mathcal{S}$. The weight in front of the marginal contribution takes into account all possible orderings in which a player can join a coalition of its respective size.

The Shapley value has a number of useful properties; in fact, it has been shown that the Shapley value is the only value that satisfies these properties [237].

Property 1: Additivity For any two games with M players and respective value functions v and u, let the combined payout for the sum game be $v + u$. Then the Shapley values for the combined game are the sum of the values of the separate games:

$$\phi_i(v + w) = \phi_i(v) + \phi_i(w), \forall i \in \mathcal{M} \tag{D.2}$$

Property 2: Efficiency The total payoff is distributed completely among all players.

$$\sum_{i=0}^{M} \phi_i = v(\mathcal{M}) \tag{D.3}$$

Property 3: Symmetry If $\forall \mathcal{S} \subseteq \mathcal{M} \setminus \{i, j\}, v(\mathcal{S} \cup \{i\}) = v(\mathcal{S} \cup \{j\})$, then $\phi_i = \phi_j$. If two players contribute equally to all possible coalitions, then their Shapley values are the same.

Property 4: Dummy If $\forall \mathcal{S} \subseteq \mathcal{M} \setminus \{i\}, v(\mathcal{S} \cup \{i\}) = v(\mathcal{S})$, then $\phi_i = 0$. If the contribution of a player to all possible coalitions is 0 then the corresponding Shapley value is 0.

D.2 Shapley Values for Machine Learning

To make the connection with machine learning, consider the players to be the input features and the value function to be the model itself. Specifically, let $f : \mathbb{R}^{\mathbb{M}} \to \mathbb{R}$ be the machine learning model that accepts M input features $x = \{x_1, \ldots, x_M\}$. Then the output of the model is distributed among all features following the efficiency property Eq. (D.3) where the Shapley value ϕ_0 corresponds to the expectation of the model over the input data distribution:

$$f(x) = \phi_0 + \sum_{i=1}^{M} \phi_i = \mathbb{E}_{\boldsymbol{x}}[f(x)] + \sum_{i=1}^{M} \phi_i \tag{D.4}$$

In order to calculate the Shapley values ϕ, it is necessary to be able to evaluate the value function for any subset of features $v(\boldsymbol{x}_{\mathcal{S}})$. To do so, [114] proposed to use the conditional expectation of the model:

$$v(\boldsymbol{x}_{\mathcal{S}}) = \mathbb{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x}) \mid \boldsymbol{x}_{\mathcal{S}}\right] \tag{D.5}$$

Where $\boldsymbol{x}_{\mathcal{S}}$ indicates the subset of feature values of the input vector $\boldsymbol{x}$ for which to evaluate v.

In most cases, calculating the exact Shapley values is computationally infeasible. In fact, it is an NP-hard problem for the general case [48]. In specific cases, the Shapley value calculations can make use of the structure of the machine learning model to efficiently calculate the exact values for tree-based models [115] or approximations for neural network models [115]. In other cases, Shapley value approximations require additional assumptions such as feature independence and model linearity [114], or use different sampling techniques [131].

Shapley values for feature embeddings

Let $f_e(x_i) : \mathbb{R} \to \mathbb{R}^k$ be the function defining the embedding which maps a one-dimensional feature value x_i to a k-dimensional feature vector $\boldsymbol{e_i} \in \mathbb{R}^k$. Then the Shapley value ϕ_i corresponding to the original feature x_i is *represented* by the k Shapley values ϕ^* of the k-dimensional embedding. A reasonable way to represent the feature contribution of the original feature is then to sum up the Shapley values of the embedding:

$$\phi_i \approx \sum_{j=1}^{k} \phi_j^* \tag{D.6}$$

It is important to note that the sum of the Shapley values of the embeddings is not necessarily equal to the Shapley value of the original feature. In fact, the Shapley value for the original feature, in general, does not exist because it is not defined for the machine learning model (the value function) that has been trained on the embedded features.

D.3 General Practitioner Data used during the Experiments

The data used during the experiments consists of resource use data from general practitioners. This section contains general aggregated statistics to provide additional information concerning the data used during the experiments.

A sample for a practitioner consists of a single performed action that corresponds to a procedure code. In Belgium, such a code is defined by the RIZIV [170] and is simply called a 'nomenclature code' [169]. For example, a patient's visit to a general practitioner is a single sample. If the GP then performs an additional procedure,for example, a patient might need an electrocardiogram performed, then this is an additional sample. If this visit takes place in the evening, then the GP is allowed to add a surcharge, which becomes another sample, etc. As each sample corresponds to a single code, the number of samples per practitioner can vary greatly, depending on the tasks performed by the GP.

Figure D.1 shows the distribution of the number of samples present for each practitioner. On the left, the distribution of all practitioners can be seen, while on the right, the distribution of the known anomalous practitioners can be observed. In total, the data set contains 11851 different general practitioners, of which 38 have been flagged as having anomalous behavior. The data of the anomalous practitioners consists of close to 1% of all available data samples before aggregation. After aggregating the data per practitioner, 0.3% of the data is labeled as anomalous.

Figure D.2 shows the distribution of the cardinality of the categorical features for all practitioners and the anomalous practitioners. The large peak in the middle represents the nomenclature code, which has the highest cardinality (159155 unique values for the full data set). The features with the lowest cardinality contain only 2 unique values. It is a simplistic representation of the distribution of the categorical data; however, it does show that the data of the anomalous GPs is diverse and similar to the distribution of the complete data set.

The feature indicating the nomenclature codes is represented in Figure D.3 by the normalized counts of the categorical values. This feature is one of the most important as it represents the actual action of a practitioner. The peak on the left side contains common nomenclature codes, such as the visit of a patient to the GP's office, while the right side contains less common codes. A code can be rare because it involves an uncommon action (for a GP) or because it is very specific. An example of a very specific code is the treatment of a fracture of a particular bone. Figure D.3 further shows that the data of the GPs with anomalous behavior contain diverse actions, similar to the distribution of the

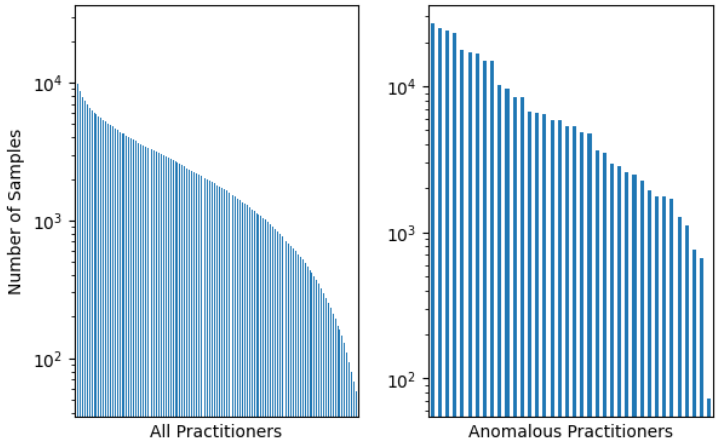


Figure D.1: The distribution of the number of samples for each practitioner (in log scale) of all practitioners (left) and the known anomalous practitioners(right).

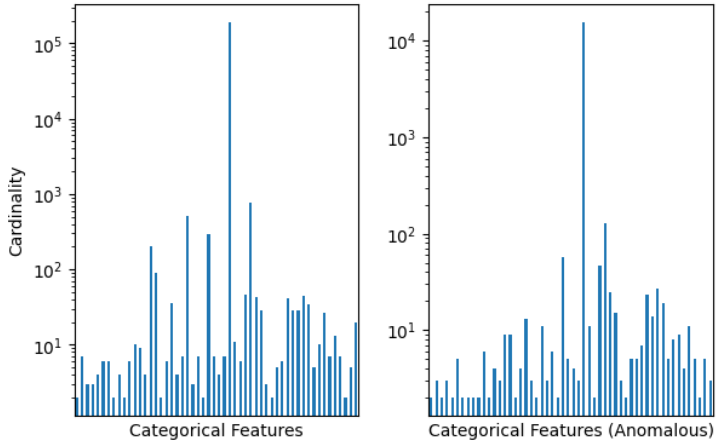


Figure D.2: The cardinality of the categorical features for all practitioners (left) and the anomalous practitioners (right).

complete data, particularly for the common actions. This further indicates that there is often no single obvious indication of anomalous behavior of a GP, which shows the need for advanced machine learning and explainability techniques.

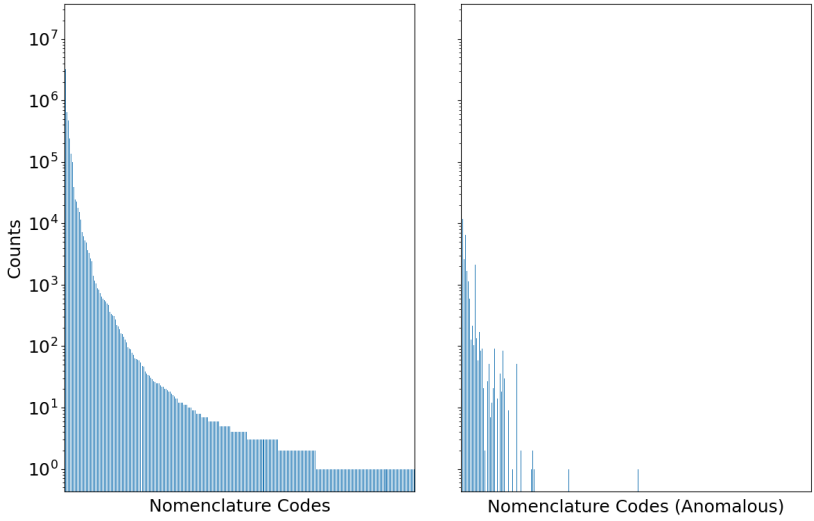


Figure D.3: The distribution of the nomenclature codes for all practitioners (left) and the anomalous practitioners (right).

Bibliography

[1] N. L. Afanador, A. Smolinska, T. N. Tran, and L. Blanchet. "Unsupervised random forest: a tutorial with case studies". In: *Journal of Chemometrics* 30.5 (2016), pp. 232–241. DOI: https://doi.org/10.1002/cem.2790.

[2] M. Ahmed, A. N. Mahmood, and M. R. Islam. "A survey of anomaly detection techniques in financial domain". In: *Future Generation Computer Systems* 55 (2016), pp. 278–288. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2015.01.001. URL: http://www.sciencedirect.com/science/article/pii/S0167739X15000023.

[3] C. M. Alaíz, M. Fanuel, and J. A. K. Suykens. "Convex Formulation for Kernel PCA and Its Use in Semisupervised Learning". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.8 (2018), pp. 3863–3869. DOI: 10.1109/TNNLS.2017.2709838.

[4] A. Alaoui and M. W. Mahoney. "Fast randomized kernel ridge regression with statistical guarantees". In: *Neural Information Processing Systems*. 2015, pp. 775–783.

[5] F. Almeida and G. Xexéo. "Word Embeddings: A Survey". In: *CoRR* abs/1901.09069 (2019). arXiv: 1901.09069. URL: http://arxiv.org/abs/1901.09069.

[6] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017.

[7] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. "Generalization and equilibrium in generative adversarial nets (gans)". In: *International conference on machine learning*. PMLR. 2017, pp. 224–232.

[8] S. Arora and Y. Zhang. "Do gans actually learn the distribution? an empirical study". In: *arXiv preprint arXiv:1706.08224* (2017).

[9] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. Toronto: ACM Press / Addison-Wesley, 1999. ISBN: 0-201-39829-X.

[10] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells. "Isolation-based anomaly detection using nearest-neighbor ensembles". In: *Computational Intelligence* 34.4 (2018), pp. 968–998. DOI: https://doi.org/10.1111/coin.12156.

[11] R. A. Bauder and T. M. Khoshgoftaar. "A Probabilistic Programming Approach for Outlier Detection in Healthcare Claims". In: *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Dec. 2016, pp. 347–354. DOI: 10.1109/ICMLA.2016.0063.

[12] R. A. Bauder, T. M. Khoshgoftaar, A. Richter, and M. Herland. "Predicting Medical Provider Specialties to Detect Anomalous Insurance Claims". In: *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. Nov. 2016, pp. 784–790. DOI: 10.1109/ICTAI.2016.0123.

[13] R. A. Bauder and T. M. Khoshgoftaar. "Multivariate outlier detection in medicare claims payments applying probabilistic programming methods". In: *Health Services and Outcomes Research Methodology* 17.3 (Dec. 2017), pp. 256–289. ISSN: 1572-9400. DOI: 10.1007/s10742-017-0172-1. URL: https://doi.org/10.1007/s10742-017-0172-1.

[14] R. A. Bauder, R. C. D. Rosa, and T. M. Khoshgoftaar. "Identifying medicare provider fraud with unsupervised machine learning". In: *Proceedings - 2018 IEEE 19th International Conference on Information Reuse and Integration for Data Science, IRI 2018* (2018), pp. 285–292. DOI: 10.1109/IRI.2018.00051.

[15] Y. Bengio, A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[16] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155.

[17] W. Bergsma. "A bias-correction for Cramér's V and Tschuprow's T". In: *Journal of the Korean Statistical Society* 42.3 (2013), pp. 323–328. ISSN: 1226-3192.

[18] R. Bhatia, T. Jain, and Y. Lim. "On the Bures–Wasserstein distance between positive definite matrices". In: *Expositiones Mathematicae* 37.2 (2019), pp. 165–191.

[19] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton. "Demystifying MMD GANs". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2018.

[20] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. "Enriching Word Vectors with Subword Information". In: *TACL* 5 (2017), pp. 135–146.

[21] T. Bolukbasi, K. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai. "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 4349–4357.

[22] T. Boyce, C. Brown, et al. "Economic and social impacts and benefits of health systems." In: *Economic and social impacts and benefits of health systems.* (2019).

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. "LOF: Identifying Density-Based Local Outliers". In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808. DOI: 10.1145/335191.335388.

[24] Bryan Lim and Sercan Ö. Arik and Nicolas Loeff and Tomas Pfister. "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting". In: *International Journal of Forecasting* 37.4 (2021), pp. 1748–1764. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2021.03.012.

[25] A. T. Bui, J.-K. Im, D. W. Apley, and G. C. Runger. "Projection-Free Kernel Principal Component Analysis for Denoising". In: *Neurocomputing* (2019). ISSN: 0925-2312.

[26] J. A. Bullinaria and J. P. Levy. "Extracting semantic representations from word co-occurrence statistics: A computational study". In: *Behavior Research Methods* 39.3 (Aug. 2007), pp. 510–526. ISSN: 1554-3528.

[27] L. M. Candanedo, V. Feldheim, and D. Deramaix. "Data driven prediction models of energy use of appliances in a low-energy house". In: *Energy and Buildings* 140 (2017), pp. 81–97. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2017.01.083.

[28] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. "Machine learning and the physical sciences". In: *Rev. Mod. Phys.* 91 (4 Dec. 2019), p. 045002. DOI: 10.1103/RevModPhys.91.045002. URL: https://link.aps.org/doi/10.1103/RevModPhys.91.045002.

[29] Centers for Medicare & Medicaid Services. *European health care expenditure*. URL: https://www.cms.gov/data-research/statistics-trends-and-reports/national-health-expenditure-data/historical (visited on 01/19/2023).

[30] C. Challu, K. G. Olivares, B. N. Oreshkin, F. Garza, M. Mergenthaler-Canseco, and A. Dubrawski. *N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting*. 2022. DOI: 10.48550/ARXIV.2201.12886.

[31] V. Chandola, S. R. Sukumar, and J. C. Schryver. "Knowledge discovery from massive healthcare claims data". In: *KDD*. 2013.

[32] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. "Mode Regularized Generative Adversarial Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[33] Y. Cho and L. Saul. "Kernel Methods for Deep Learning". In: *Advances in Neural Information Processing Systems*. Ed. by Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta. Vol. 22. Curran Associates, Inc., 2009.

[34] CMS. *HCPCS - Centers for Medicare and Medicaid Services*. URL: https://www.cms.gov/medicare/coding/medhcpcsgeninfo (visited on 03/19/2023).

[35] H. Cramer. *Mathematical methods of statistics*. English. Princeton University Press Princeton, 1946, xvi, 575 p.

[36] S. Curi, K. Levy, S. Jegelka, A. Krause, et al. "Adaptive Sampling for Stochastic Risk-Averse Learning". In: *Neural Information Processing Systems* (2020).

[37] A. Damianou and N. D. Lawrence. "Deep Gaussian Processes". In: *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*. Ed. by C. M. Carvalho and P. Ravikumar. Vol. 31. Proceedings of Machine Learning Research. Scottsdale, Arizona, USA: PMLR, Apr. 2013, pp. 207–215.

[38] S. Dasgupta and A. Gupta. "An Elementary Proof of a Theorem of Johnson and Lindenstrauss". In: *Random Struct. Algorithms* 22.1 (Jan. 2003), pp. 60–65.

[39] M. J. Davis. "Contrast Coding in Multiple Regression Analysis: Strengths, Weaknesses, and Utility of Popular Coding Structures". In: *Journal of Data Science* 8.1 (Jan. 2010), pp. 61–73. ISSN: 1683-8602.

[40] H. De Meulemeester and B. De Moor. "Unsupervised Embeddings for Categorical Variables". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207703.

[41] H. De Meulemeester, F. De Smet, J. van Dorst, E. Derroitte, and B. De Moor. "Explainable Unsupervised Anomaly Detection for Healthcare Insurance Data". Internal Report 23-129, ESAT-STADIUS, KU Leuven (Leuven, Belgium), 2023.

[42] H. De Meulemeester, J. Schreurs, M. Fanuel, B. De Moor, and J. A. Suykens. “The Bures Metric for Generative Adversarial Networks”. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (2021).

[43] H. De Meulemeester, J. Schreurs, M. Fanuel, B. De Moor, and J. A. Suykens. “The Bures metric for generative adversarial networks”. In: *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*. Springer. 2021, pp. 52–66.

[44] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. “Indexing by Latent Semantic Analysis”. In: *JASIS* 41.6 (1990), pp. 391–407.

[45] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. “Indexing by Latent Semantic Analysis”. In: *JASIS* 41.6 (1990), pp. 391–407.

[46] A. Defazio, F. R. Bach, and S. Lacoste-Julien. “SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives”. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2014, pp. 1646–1654.

[47] M. Deisenroth and J. W. Ng. “Distributed gaussian processes”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1481–1490.

[48] X. Deng and C. H. Papadimitriou. “On the Complexity of Cooperative Solution Concepts”. In: *Mathematics of Operations Research* 19.2 (1994), pp. 257–266. ISSN: 0364765X, 15265471. (Visited on 02/01/2024).

[49] A. B. Dieng, F. J. R. Ruiz, D. M. Blei, and M. K. Titsias. “Prescribed Generative Adversarial Networks”. In: *arxiv:1910.04302* (2020).

[50] M. Diesendruck, E. R. Elenberg, R. Sen, G. W. Cole, S. Shakkottai, and S. A. Williamson. “Importance weighted generative networks”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2019, pp. 249–265.

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

[52] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[53] EHFCN. *The EHFCN Waste Typology Matrix*. URL: http://www.ehfcn.org/what-is-fraud/ehfcn-waste-typology-matrix/ (visited on 01/24/2023).

[54] M. Elfeki, C. Couprie, M. Riviere, and M. Elhoseiny. "GDPP: Learning Diverse Generations using Determinantal Point Processes". In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019.

[55] Y. Engel, S. Mannor, and R. Meir. "The kernel recursive least-squares algorithm". In: *IEEE Transactions on Signal Processing* 52.8 (2004), pp. 2275–2285. DOI: 10.1109/TSP.2004.830985.

[56] D. Erhan, Y. Bengio, A. C. Courville, P. Manzagol, P. Vincent, and S. Bengio. "Why Does Unsupervised Pre-training Help Deep Learning?" In: *Journal of Machine Learning Research* 11 (2010), pp. 625–660.

[57] Eurostat. *European health care expenditure*. URL: https://ec.europa.eu/eurostat/databrowser/view/hlth%5C_sha11%5C_hf/default/table (visited on 01/19/2023).

[58] V. Feldman. "Does Learning Require Memorization? A Short Tale about a Long Tail". In: *52nd Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2020. 2020, pp. 954–959.

[59] J. H. Friedman. "Multivariate Adaptive Regression Splines". In: *Ann. Statist.* 19.1 (Mar. 1991), pp. 1–67. DOI: 10.1214/aos/1176347963. URL: https://doi.org/10.1214/aos/1176347963.

[60] M. Gelbrich. "On a Formula for the L2 Wasserstein Metric between Measures on Euclidean and Hilbert Spaces". In: *Mathematische Nachrichten* 147.1 (1990), pp. 185–203.

[61] A. Ghosh, V. Kulharia, V. P. Namboodiri, P. H. Torr, and P. K. Dokania. "Multi-Agent Diverse Generative Adversarial Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[62] I. Goodfellow. "Nips 2016 tutorial: Generative adversarial networks". In: *arXiv preprint arXiv:1701.00160* (2016).

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

[64] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Neural information processing systems*. 2014, pp. 2672–2680.

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

[66] A. Goodge, B. Hooi, S.-K. Ng, and W. S. Ng. "LUNAR: Unifying Local Outlier Detection Methods via Graph Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.6 (June 2022), pp. 6737–6745. DOI: 10.1609/aaai.v36i6.20629. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20629.

[67] Google. *Bard*. URL: https://bard.google.com/?hl=en (visited on 06/19/2023).

[68] E. Grave, T. Mikolov, A. Joulin, and P. Bojanowski. "Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*. Association for Computational Linguistics, 2017, pp. 427–431.

[69] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. "Improved training of Wasserstein GANs". In: *Advances in neural information processing systems 31*. 2017.

[70] C. Guo and F. Berkhahn. "Entity Embeddings of Categorical Variables". In: *CoRR* abs/1604.06737 (2016). arXiv: 1604.06737. URL: http://arxiv.org/abs/1604.06737.

[71] W. Guo, J. Wang, and S. Wang. "Deep Multimodal Representation Learning: A Survey". In: *IEEE Access* 7 (2019), pp. 63373–63394. DOI: 10.1109/ACCESS.2019.2916887.

[72] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[73] S. He, Y. Li, Y. Feng, S. Ho, S. Ravanbakhsh, W. Chen, and B. Póczos. "Learning to predict the cosmological structure formation". In: *Proceedings of the National Academy of Sciences* 116.28 (2019), pp. 13825–13832. DOI: 10.1073/pnas.1821458116. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1821458116. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1821458116.

[74] M. Herland, T. M. Khoshgoftaar, and R. A. Bauder. "Big Data fraud detection using multiple medicare data sources". In: *Journal of Big Data* (Sept. 2018). ISSN: 2196-1115. DOI: 10.1186/s40537-018-0138-3. URL: https://doi.org/10.1186/s40537-018-0138-3.

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

[76] G. E. Hinton. "A Practical Guide to Training Restricted Boltzmann Machines". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by G. Montavon, G. B. Orr, and K.-R. Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 599–619. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_32. URL: https://doi.org/10.1007/978-3-642-35289-8_32.

[77] G. E. Hinton. "Training Products of Experts by Minimizing Contrastive Divergence". In: *Neural Comput.* 14.8 (Aug. 2002), pp. 1771–1800. ISSN: 0899-7667. DOI: 10.1162/089976602760128018.

[78] G. E. Hinton. "What Kind of a Graphical Model is the Brain?" In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. IJCAI'05. Edinburgh, Scotland: Morgan Kaufmann Publishers Inc., 2005, pp. 1765–1775.

[79] P. Honeine and C. Richard. "Preimage Problem in Kernel-Based Machine Learning". In: *IEEE Signal Processing Magazine* 28.2 (Mar. 2011), pp. 77–88. ISSN: 1053-5888.

[80] Y. Hong and R. A. Horn. "The Jordan Canonical Form of a Product of a Hermitian and a Positive Semidefinite Matrix". In: *Linear Algebra and its Applications* 147 (1991), pp. 373–386.

[81] L. Houthuys and J. A. Suykens. "Tensor-based restricted kernel machines for multi-view classification". In: *Information Fusion* 68 (2021), pp. 54–66. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2020.10.022.

[82] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. New York: Springer, 2013.

[83] D. Jurafsky and J. H. Martin. *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009. ISBN: 0131873210.

[84] M. Kallas, P. Honeine, C. Francis, and H. Amoud. "Kernel autoregressive models using Yule-Walker equations". In: *Signal Processing* 93.11 (2013), pp. 3053–3061. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2013.03.032. URL: https://www.sciencedirect.com/science/article/pii/S0165168413001242.

[85] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[86] H. Kaya, P. Tüfekci, and E. Uzun. "Predicting CO and NOx emissions from gas turbines: Novel data and a benchmark PEMS". In: *Turkish Journal of Electrical Engineering and Computer Sciences* 27.6 (Nov. 2019), pp. 4783–4796. DOI: 10.3906/ELK-1807-87.

[87] Y. Kim, Y. Jernite, D. A. Sontag, and A. M. Rush. "Character-Aware Neural Language Models". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. 2016, pp. 2741–2749.

[88] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015.

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

[90] D. P. Kingma and M. Welling. "Auto-encoding Variational Bayes". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014.

[91] T. G. Kolda. "Limited-Memory Matrix Methods with Applications". PhD thesis. University of Maryland, 1997.

[92] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 831–838.

[93] H.-P. Kriegel, M. Schubert, and A. Zimek. "Angle-Based Outlier Detection in High-Dimensional Data". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. Las Vegas, Nevada, USA: Association for Computing Machinery, 2008, pp. 444–452. ISBN: 9781605581934. DOI: 10.1145/1401890.1401946. URL: https://doi.org/10.1145/1401890.1401946.

[94] J. T. Kwok and I. W.-H. Tsang. "The pre-image problem in kernel methods". In: *IEEE Transactions on Neural Networks* 15 (2003), pp. 1517–1525.

[95] N. Lawrence. "Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models". In: *Journal of Machine Learning Research* 6.60 (2005), pp. 1783–1816.

[96] Q. V. Le and T. Mikolov. "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. 2014, pp. 1188–1196.

[97] R. Lebret and R. Collobert. “Word Embeddings through Hellinger PCA”. In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*. The Association for Computer Linguistics, 2014, pp. 482–490.

[98] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. “Handwritten digit recognition with a back-propagation network”. In: *Advances in neural information processing systems* 2 (1989).

[99] Y. LeCun and C. Cortes. *MNIST handwritten digit database*. 2010. URL: http://yann.lecun.com/exdb/mnist/.

[100] Y. LeCun, K. Kavukcuoglu, and C. Farabet. “Convolutional networks and applications in vision”. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010, pp. 253–256. DOI: 10.1109/ISCAS.2010.5537907.

[101] K. S. Lee, N.-T. Tran, and N.-M. Cheung. “InfoMax-GAN: Mutual Information Maximization for Improved Adversarial Image Generation”. In: *NeurIPS 2019 Workshop on Information Theory and Machine Learning*. 2019.

[102] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Poczos. “MMD GAN: Towards Deeper Understanding of Moment Matching Network”. In: *Advances in Neural Information Processing Systems 30*. 2017.

[103] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago, T. Hubert, P. Choy, C. de Masson d’Autume, I. Babuschkin, X. Chen, P.-S. Huang, J. Welbl, S. Gowal, A. Cherepanov, J. Molloy, D. J. Mankowitz, E. S. Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, and O. Vinyals. “Competition-level code generation with AlphaCode”. In: *Science* 378.6624 (2022), pp. 1092–1097. DOI: 10.1126/science.abq1158. URL: https://www.science.org/doi/abs/10.1126/science.abq1158.

[104] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu. “COPOD: Copula-Based Outlier Detection”. In: *2020 IEEE International Conference on Data Mining (ICDM)*. 2020, pp. 1118–1123. DOI: 10.1109/ICDM50108.2020.00135.

[105] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. Chen. “ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions”. In: *IEEE Transactions on Knowledge and Data Engineering* (2022), pp. 1–1. DOI: 10.1109/TKDE.2022.3159580.

[106] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. "A Structured Self-attentive Sentence Embedding". In: *CoRR* abs/1703.03130 (2017). arXiv: 1703.03130. URL: http://arxiv.org/abs/1703.03130.

[107] Z. Lin, A. Khetan, G. Fanti, and S. Oh. "PacGAN: The power of two samples in generative adversarial networks". In: *Advances in Neural Information Processing Systems 31*. 2018.

[108] Z. Lin, A. Khetan, G. Fanti, and S. Oh. "PacGAN: The power of two samples in generative adversarial networks". In: *Neural Information Processing Systems*. Vol. 31. 2018, pp. 1498–1507.

[109] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez. "A survey on deep learning in medical image analysis". In: *Medical Image Analysis* 42 (2017), pp. 60–88. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2017.07.005. URL: https://www.sciencedirect.com/science/article/pii/S1361841517301135.

[110] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[111] P. M. Long and R. A. Servedio. "Restricted boltzmann machines are hard to approximately evaluate or simulate". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 703–710. ISBN: 9781605589077.

[112] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. "Are gans created equal? a large-scale study". In: *Advances in neural information processing systems*. 2018, pp. 700–709.

[113] K. Lund and C. Burgess. "Producing high-dimensional semantic spaces from lexical co-occurrence". In: *Behavior Research Methods, Instruments, & Computers* 28.2 (June 1996), pp. 203–208. ISSN: 1532-5970.

[114] S. M. Lundberg and S.-I. Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[115] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. "From local explanations to global understanding with explainable AI for trees". In: *Nature Machine Intelligence* 2.1 (2020), pp. 2522–5839.

[116] P. Ma, M. Mahoney, and B. Yu. "A Statistical Perspective on Algorithmic Leveraging". In: *31st International Conference on Machine Learning*. Vol. 32. Proceedings of Machine Learning Research 1. 2014, pp. 91–99.

[117] L. van der Maaten and G. Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[118] D. J. Mankowitz, A. Michi, A. Zhernov, M. Gelmi, M. Selvi, C. Paduraru, E. Leurent, S. Iqbal, J.-B. Lespiau, A. Ahern, T. Köppe, K. Millikin, S. Gaffney, S. Elster, J. Broshear, C. Gamble, K. Milan, R. Tung, M. Hwang, T. Cemgil, M. Barekatain, Y. Li, A. Mandhane, T. Hubert, J. Schrittwieser, D. Hassabis, P. Kohli, M. Riedmiller, O. Vinyals, and D. Silver. “Faster sorting algorithms discovered using deep reinforcement learning”. In: *Nature* 618.7964 (June 2023), pp. 257–263. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06004-9. URL: https://doi.org/10.1038/s41586-023-06004-9.

[119] A. Marino and L. Lorenzoni. “The impact of technological advancements on health spending”. In: 113 (2019). DOI: https://doi.org/https://doi.org/10.1787/fa3bab05-en. URL: https://www.oecd-ilibrary.org/content/paper/fa3bab05-en.

[120] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: https://www.tensorflow.org/.

[121] E. Massart and P.-A. Absil. “Quotient Geometry with Simple Geodesics for the Manifold of Fixed-Rank Positive-Semidefinite Matrices”. In: *SIAM Journal on Matrix Analysis and Applications* 41.1 (2020), pp. 171–198.

[122] L. McInnes, J. Healy, and J. Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *preprint arXiv:1802.03426* (2018).

[123] J. Mercer. “Functions of Positive and Negative Type, and Their Connection the Theory of Integral Equations”. In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 209.441-458 (Jan. 1909), pp. 415–446.

[124] L. Mescheder, S. Nowozin, and A. Geiger. “The Numerics of GANs”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 1823–1833.

[125] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. "Unrolled Generative Adversarial Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[126] D. Micci-Barreca. "A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems". In: *SIGKDD Explorations* 3.1 (2001), pp. 27–32.

[127] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Mullers. "Fisher discriminant analysis with kernels". In: *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*. 1999, pp. 41–48. DOI: 10.1109/NNSP.1999.788121.

[128] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. "Kernel PCA and De-noising in Feature Spaces". In: *Advances in Neural Information Processing Systems*. MIT Press, 1999, pp. 536–542.

[129] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013. URL: http://arxiv.org/abs/1301.3781.

[130] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 2013, pp. 3111–3119.

[131] R. Mitchell, J. Cooper, E. Frank, and G. Holmes. "Sampling permutations for Shapley value estimation". In: 23.1 (2022). ISSN: 1532-4435.

[132] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. "Spectral normalization for generative adversarial networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2018.

[133] Y. Mroueh, T. Sercu, and V. Goel. "McGan: Mean and Covariance Feature Matching GAN". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017.

[134] K. .-. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. "Predicting time series with support vector machines". In: *Artificial Neural Networks — ICANN'97*. Ed. by W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 999–1004. ISBN: 978-3-540-69620-9.

[135] C. Musco and C. Musco. "Recursive sampling for the nystrom method". In: *Neural Information Processing Systems*. 2017, pp. 3833–3845.

[136] T. Nguyen, T. Le, H. Vu, and D. Phung. "Dual Discriminator Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017, pp. 2670–2680.

[137] M. Nissim, R. van Noord, and R. van der Goot. "Fair is Better than Sensational: Man is to Doctor as Woman is to Doctor". In: *CoRR* abs/1905.09866 (2019). arXiv: 1905.09866. URL: http://arxiv.org/abs/1905.09866.

[138] J. H. Oh, M. Pouryahya, A. Iyer, A. P. Apte, J. O. Deasy, and A. Tannenbaum. "A novel kernel Wasserstein distance on Gaussian measures: An application of identifying dental artifacts in head and neck computed tomography". In: *Computers in Biology and Medicine* 120 (2020), p. 103731.

[139] M. Olave, V. Rajkovic, and M. Bohanec. "An application for admission in public school systems". In: *Expert Systems in Public Administration,* 1989, pp. 145–160.

[140] OpenAI. *ChatGPT*. URL: https://openai.com/blog/chatgpt (visited on 06/19/2023).

[141] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[142] OpenAI, : C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019. arXiv: 1912.06680 [cs.LG].

[143] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *International Conference on Learning Representations*. 2020.

[144] T. Osogami. "Boltzmann Machines and Energy-Based Models". In: *arXiv:1708.06008 [cs]* (2019). arXiv: 1708.06008 [cs].

[145] T. Osogami. "Boltzmann Machines for Time-Series". In: *arXiv:1708.06004 [cs]* (2019). arXiv: 1708.06004 [cs].

[146] S. Oymak, B. Recht, and M. Soltanolkotabi. "Isometric sketching of any set via the Restricted Isometry Property". In: *Information and Inference: A Journal of the IMA* 7.4 (Mar. 2018), pp. 707–726.

[147] A. Pandey, H. De Meulemeester, B. De Moor, and J. A. Suykens. "Multi-view kernel PCA for time series forecasting". In: *Neurocomputing* 554 (2023), p. 126639. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.126639.

[148] A. Pandey, H. De Meulemeester, H. De Plaen, B. De Moor, and J. Suykens. "Recurrent Restricted Kernel Machines for Time-series Forecasting". en. In: *ESANN 2022 proceedings*. Bruges (Belgium) and online event, 2022, pp. 399–404. ISBN: 978-2-87587-084-1. DOI: 10.14428/esann/2022.ES2022-120. (Visited on 11/17/2022).

[149] A. Pandey, H. De Meulemeester, H. De Plaen, B. De Moor, and J. A. K. Suykens. "Recurrent Restricted Kernel Machines for Time-series Forecasting". In: *Proceedings of ESANN 2022* (2022).

[150] A. Pandey, M. Fanuel, J. Schreurs, and J. A. K. Suykens. *Disentangled Representation Learning and Generation with Manifold Optimization*. 2020. DOI: 10.48550/ARXIV.2006.07046. arXiv: 2006.07046 [cs, stat]. URL: https://arxiv.org/abs/2006.07046.

[151] A. Pandey, M. Fanuel, J. Schreurs, and J. A. K. Suykens. "Disentangled Representation Learning and Generation with Manifold Optimization". In: *To appear in Neural Computation. CoRR* abs/2006.07046 (2020). DOI: 10.48550/ARXIV.2006.07046. arXiv: 2006.07046. URL: https://arxiv.org/abs/2006.07046.

[152] A. Pandey, J. Schreurs, and J. A. K. Suykens. "Robust Generative Restricted Kernel Machines using Weighted Conjugate Feature Duality". In: *proceedings of the Sixth International Conference on Machine Learning, Optimization, and Data Science (LOD)* (Siena – Tuscany, Italy). Siena – Tuscany, Italy, 2020.

[153] A. Pandey, J. Schreurs, and J. A. Suykens. "Generative Restricted Kernel Machines: A framework for multi-view generation and disentangled feature learning". In: *Neural Networks* 135 (2021), pp. 177–191.

[154] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[155] J. Pennington, R. Socher, and C. D. Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2014, pp. 1532–1543.

[156] J. L. Perols, R. M. Bowen, C. Zimmermann, and B. Samba. "Finding Needles in a Haystack: Using Data Analytics to Improve Fraud Prediction." In: *Accounting Review* 92.2 (2017), pp. 221–245. ISSN: 00014826.

[157] T. Pevný. “Loda: Lightweight on-line detector of anomalies”. In: *Machine Learning* 102.2 (Feb. 2016), pp. 275–304. ISSN: 1573-0565. DOI: 10.1007/s10994-015-5521-0. URL: https://doi.org/10.1007/s10994-015-5521-0.

[158] R. Peyre. “Comparison between W_2 distance and $\dot{H}^{-1}$ norm, and Localization of Wasserstein distance”. In: *ESAIM: COCV* 24.4 (2018), pp. 1489–1501.

[159] G. Peyré, M. Cuturi, et al. “Computational Optimal Transport”. In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607.

[160] A. Radford, L. Metz, and S. Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016.

[161] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad. “Machine learning at the energy and intensity frontiers of particle physics”. In: *Nature* 560.7716 (Aug. 2018), pp. 41–48. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0361-2. URL: https://doi.org/10.1038/s41586-018-0361-2.

[162] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, P. Sundberg, H. Yee, K. Zhang, Y. Zhang, G. Flores, G. E. Duggan, J. Irvine, Q. Le, K. Litsch, A. Mossin, J. Tansuwan, D. Wang, J. Wexler, J. Wilson, D. Ludwig, S. L. Volchenboum, K. Chou, M. Pearson, S. Madabushi, N. H. Shah, A. J. Butte, M. D. Howell, C. Cui, G. S. Corrado, and J. Dean. “Scalable and accurate deep learning with electronic health records”. In: *npj Digital Medicine* 1.1 (May 2018), p. 18. ISSN: 2398-6352. DOI: 10.1038/s41746-018-0029-1. URL: https://doi.org/10.1038/s41746-018-0029-1.

[163] S. Ramaswamy, R. Rastogi, and K. Shim. “Efficient Algorithms for Mining Outliers from Large Data Sets”. In: *SIGMOD Rec.* 29.2 (May 2000), pp. 427–438. ISSN: 0163-5808. DOI: 10.1145/335191.335437. URL: https://doi.org/10.1145/335191.335437.

[164] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. ISBN: 026218253X.

[165] Y. Ren, J. Zhu, J. Li, and Y. Luo. “Conditional Generative Moment-Matching Networks”. In: *Advances in Neural Information Processing Systems 29*. 2016.

[166] D. J. Rezende and S. Mohamed. "Variational Inference with Normalizing Flows". In: *Proceedings of the 32th International Conference on Machine Learning (ICML)* (2015).

[167] RIZIV. *Actieplan handhaving in de gezondheidszorg*. URL: https://www.riziv.fgov.be/nl/themas/zorgkwaliteit/Paginas/doelmatigheid-gezondheidszorg.aspx (visited on 06/19/2023).

[168] RIZIV. *Health*. URL: https://riziv.fgov.be/nl/professionals/andere-professionals/ziekenfonds/Paginas/contacteer-ziekenfondsen.aspx (visited on 06/19/2023).

[169] RIZIV. *Nomenclatuur van de geneeskundige verstrekkingen - Riziv*. URL: https://www.riziv.fgov.be/nl/nomenclatuur/Paginas/default.aspx (visited on 03/19/2023).

[170] RIZIV. *Rijksinstituut voor ziekte- en invaliditeitsverzekering*. URL: https://www.riziv.fgov.be/ (visited on 01/19/2023).

[171] R. T. Rockafellar. *Conjugate duality and optimization*. SIAM, 1974.

[172] D. L. T. Rohde, L. M. Gonnerman, and D. C. Plaut. "An improved model of semantic similarity based on lexical co-occurence". In: *COMMUNICATIONS OF THE ACM* 8 (2006), pp. 627–633.

[173] M. Rostami, H. He, M. Chen, and D. Roth. "Transfer Learning via Representation Learning". In: *Federated and Transfer Learning*. Ed. by R. Razavi-Far, B. Wang, M. E. Taylor, and Q. Yang. Cham: Springer International Publishing, 2023, pp. 233–257. ISBN: 978-3-031-11748-0. DOI: 10.1007/978-3-031-11748-0_10. URL: https://doi.org/10.1007/978-3-031-11748-0_10.

[174] B. Rozemberczki, P. Scherer, O. Kiss, R. Sarkar, and T. Ferenci. *Chickenpox Cases in Hungary: a Benchmark Dataset for Spatiotemporal Signal Processing with Graph Neural Networks*. 2021. arXiv: 2102.08100 [cs.SI].

[175] A. Rudi, D. Calandriello, L. Carratino, and L. Rosasco. "On fast leverage score sampling and optimal learning". In: *Neural Information Processing Systems*. 2018, pp. 5672–5682.

[176] L. Ruff, N. Görnitz, L. Deecke, S. A. Siddiqui, R. A. Vandermeulen, A. Binder, E. Müller, and M. Kloft. "Deep One-Class Classification". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by J. G. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. Cambridge, MA: PMLR, 2018, pp. 4390–4399. URL: http://proceedings.mlr.press/v80/ruff18a.html.

[177] Y. Russac, O. Caelen, and L. He-Guelton. "Embeddings of Categorical Variables for Sequential Data in Fraud Context". In: *The International Conference on Advanced Machine Learning Technologies and Applications, AMLTA 2018, Cairo, Egypt, February 22-24, 2018*. Vol. 723. Advances in Intelligent Systems and Computing. Springer, 2018, pp. 542–552.

[178] M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. "Assessing Generative Models via Precision and Recall". In: *32nd International Conference on Neural Information Processing Systems*. 2018, pp. 5234–5243.

[179] R. Salakhutdinov and G. Hinton. "Deep Boltzmann Machines". en. In: *proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*. Vol. 5 of JMLR. PMLR, Apr. 2009, pp. 448–455.

[180] R. Salakhutdinov, A. Mnih, and G. Hinton. "Restricted Boltzmann Machines for Collaborative Filtering". en. In: *ICML '07*. Corvalis, Oregon: ACM Press, 2007, pp. 791–798.

[181] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. "Improved Techniques for Training GANs". In: *Advances in neural information processing systems 29*. 2016.

[182] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. "Estimating the Support of a High-Dimensional Distribution". In: *Neural Computation* 13.7 (July 2001), pp. 1443–1471. DOI: 10.1162/089976601750264965. URL: https://doi.org/10.1162%2F089976601750264965.

[183] B. Schölkopf, A. Smola, and K.-R. Müller. "Kernel principal component analysis". In: *International conference on artificial neural networks*. Springer. 1997, pp. 583–588.

[184] B. Schölkopf, A. Smola, and K.-R. Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem". In: *Neural Computation* 10.5 (1998), pp. 1299–1319. DOI: 10.1162/089976698300017467.

[185] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001. ISBN: 0262194759.

[186] J. Schreurs, H. De Meulemeester, M. Fanuel, B. De Moor, and J. A. Suykens. "Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks". In: *Machine Learning, Optimization, and Data Science: 7th International Conference, LOD 2021, Grasmere, UK, October 4–8, 2021, Revised Selected Papers, Part II*. Springer. 2022, pp. 466–480.

[187] J. Schreurs and J. A. K. Suykens. “Generative Kernel PCA”. In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 2018, pp. 129–134.

[188] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis. “Improved protein structure prediction using potentials from deep learning”. en. In: *Nature* 577.7792 (Jan. 2020), pp. 706–710.

[189] L. S. Shapley. *A Value for N-Person Games*. Santa Monica, CA: RAND Corporation, 1952. DOI: 10.7249/P0295.

[190] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961. URL: https://doi.org/10.1038/nature16961.

[191] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144. DOI: 10.1126/science.aar6404.

[192] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. 2017. arXiv: 1712.01815 [cs.AI].

[193] S. Sinha, H. Zhang, A. Goyal, Y. Bengio, H. Larochelle, and A. Odena. “Small-GAN: Speeding up GAN Training using Core-Sets”. In: *37th International Conference on Machine Learning*. Vol. 119. 2020, pp. 9005–9015.

[194] Y. Song and S. Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 11918–11930.

[195] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. “Veegan: Reducing Mode Collapse in GANs using Implicit Variational Learning”. In: *Advances in Neural Information Processing Systems 30*. 2017.

[196] I. Sutskever and G. Hinton. "Learning Multilevel Distributed Representations for High-Dimensional Sequences". In: *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*. Ed. by M. Meila and X. Shen. Vol. 2. Proceedings of Machine Learning Research. San Juan, Puerto Rico: PMLR, Mar. 2007, pp. 548–555.

[197] I. Sutskever, G. E. Hinton, and G. W. Taylor. "The Recurrent Temporal Restricted Boltzmann Machine". In: *Advances in Neural Information Processing Systems*. Vol. 21. 2008. URL: https://proceedings.neurips.cc/paper/2008/file/9ad6aaed513b73148b7d49f70afcfb32-Paper.pdf.

[198] J. A. K. Suykens. "Deep Restricted Kernel Machines using Conjugate Feature Duality". en. In: *Neural Computation* 29.8 (2017), pp. 2123–2163. ISSN: 0899-7667, 1530-888X.

[199] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. en. River Edge, NJ: World Scientific, Jan. 2002. ISBN: 978-981-238-151-4.

[200] J. A. K. Suykens and J. Vandewalle. "Recurrent least squares support vector machines". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 47.7 (2000), pp. 1109–1114. DOI: 10.1109/81.855471.

[201] J. A. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor. "A support vector machine formulation to PCA analysis and its kernel version". In: *IEEE Transactions on Neural Networks* 14.2 (2003), pp. 447–450.

[202] The National Alliance of Christian Mutualities. *International website / CM*. 2023. URL: https://www.cm.be/international (visited on 01/19/2023).

[203] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, et al. "Lamda: Language models for dialog applications". In: *arXiv preprint arXiv:2201.08239* (2022).

[204] D. Thornton, G. van Capelleveen, M. Poel, J. Hillegersberg, and R. Mueller. "Outlier-based health insurance fraud detection for U.S. medicaid data". In: *ICEIS 2014 - Proceedings of the 16th International Conference on Enterprise Information Systems* 2 (Jan. 2014), pp. 684–694.

[205] D. Thornton, R. M. Mueller, P. Schoutsen, and J. van Hillegersberg. "Predicting Healthcare Fraud in Medicaid: A Multidimensional Data Model and Analysis Techniques for Fraud Detection". In: *Procedia Technology* 9 (2013). CENTERIS 2013 - Conference on ENTERprise Information Systems / ProjMAN 2013 - International Conference on

Project MANagement/ HCIST 2013 - International Conference on Health and Social Care Information Systems and Technologies, pp. 1252–1264. ISSN: 2212-0173. DOI: https://doi.org/10.1016/j.protcy.2013.12.140. URL: http://www.sciencedirect.com/science/article/pii/S2212017313002946.

[206] T. Tieleman. "Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient". In: Jan. 2008, pp. 1064–1071. DOI: 10.1145/1390156.1390290.

[207] M. Titsias. "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Ed. by D. van Dyk and M. Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, Apr. 2009, pp. 567–574.

[208] I. O. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schölkopf. "Adagan: Boosting generative models". In: *Neural Information Processing Systems*. 2017, pp. 5424–5433.

[209] F. Tonin, A. Pandey, P. Patrinos, and J. A. K. Suykens. "Unsupervised Energy-based Out-of-distribution Detection using Stiefel-Restricted Kernel Machine". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533706.

[210] F. Tonin, P. Patrinos, and J. A. Suykens. "Unsupervised learning of disentangled representations in deep restricted kernel machines with orthogonality constraints". In: *Neural Networks* 142 (2021), pp. 661–679. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2021.07.023. URL: https://www.sciencedirect.com/science/article/pii/S0893608021002860.

[211] F. Tonin, Q. Tao, P. Patrinos, and J. A. Suykens. "Deep Kernel Principal Component Analysis for multi-level feature learning". In: *Neural Networks* 170 (2024), pp. 578–595. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2023.11.045.

[212] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[213] P. Travaille, R. Mueller, D. Thornton, and J. van Hillegersberg. "Electronic Fraud Detection in the U.S. Medicaid Healthcare Program: Lessons Learned from other Industries". English. In: *Proceedings of the*

17th Americas Conference on Information Systems (AMCIS). United States: Association for Information Systems, Aug. 2011, pp. 1–11.

[214] A. Tripp, E. Daxberger, and J. M. Hernández-Lobato. "Sample-efficient optimization in the latent space of deep generative models via weighted retraining". In: *Neural Information Processing Systems* 33 (2020).

[215] G. van Capelleveen, M. Poel, R. M. Mueller, D. Thornton, and J. van Hillegersberg. "Outlier detection in healthcare fraud: A case study in the Medicaid dental domain". In: *International Journal of Accounting Information Systems* 21 (2016), pp. 18–31. ISSN: 1467-0895. DOI: https://doi.org/10.1016/j.accinf.2016.04.001. URL: https://www.sciencedirect.com/science/article/pii/S1467089515300324.

[216] S. Van Vaerenbergh, J. Via, and I. Santamaria. "A Sliding-Window Kernel RLS Algorithm and Its Application to Nonlinear Channel Identification". In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. Vol. 5. 2006, pp. V–V. DOI: 10.1109/ICASSP.2006.1661394.

[217] S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaria. "Kernel Recursive Least-Squares Tracker for Time-Varying Regression". In: *Neural Networks and Learning Systems, IEEE Transactions on* 23 (Aug. 2012), pp. 1313–1326. DOI: 10.1109/TNNLS.2012.2200500.

[218] S. Van Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe. "Fixed-budget kernel recursive least-squares". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 1882–1885. DOI: 10.1109/ICASSP.2010.5495350.

[219] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. ISBN: 0-387-94559-8.

[220] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), pp. 350–354. ISSN: 0028-0836. DOI: 10.1038/s41586-019-1724-z.

[221] W. Wang, Y. Sun, and S. Halgamuge. "Improving MMD-GAN training with repulsive loss function". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[222] A. Weigend and N. Gershenfeld. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, 1993.

[223] M. Welling, M. Rosen-zvi, and G. E. Hinton. "Exponential Family Harmoniums with an Application to Information Retrieval". In: *Advances in Neural Information Processing Systems*. Ed. by L. Saul, Y. Weiss, and L. Bottou. Vol. 17. MIT Press, 2004.

[224] C. A. W. C. A. Wendorf. "Primer on Multiple Regression Coding: Common Forms and the Additional Case of Repeated Contrasts". In: *Understanding Statistics* 3.1 (2004), pp. 47–57.

[225] J. Weston, B. Schölkopf, and G. Bakir. "Learning to Find Pre-Images". In: *Advances in Neural Information Processing Systems*. Vol. 16. 2003. URL: https://proceedings.neurips.cc/paper/2003/file/ac1ad983e08ad3304a97e147f522747e-Paper.pdf.

[226] WHO Europe. *The Tallinn Charter: Health Systems for Health and Wealth*. URL: https://iris.who.int/bitstream/handle/10665/349648/WHO-EURO-2008-4229-43988-62011-eng.pdf (visited on 06/19/2023).

[227] M. A. Wiering and L. R. Schomaker. "Multi-Layer Support Vector Machines". In: *Regularization, optimization, kernels, and support vector machines*. Ed. by J. A. K. Suykens, M. Signoretto, and A. Argyriou. Chapman and Hall/CRC, 2014, pp. 457–476.

[228] C. Williams and C. Rasmussen. "Gaussian Processes for Regression". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky, M. Mozer, and M. Hasselmo. Vol. 8. MIT Press, 1995.

[229] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. "Deep Kernel Learning". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Ed. by A. Gretton and C. C. Robert. Vol. 51. Proceedings of Machine Learning Research. Cadiz, Spain: PMLR, May 2016, pp. 370–378. URL: https://proceedings.mlr.press/v51/wilson16.html.

[230] D. Winant, J. Schreurs, and J. A. K. Suykens. "Latent Space Exploration Using Generative Kernel PCA". In: *Artificial Intelligence and Machine Learning*. Ed. by B. Bogaerts, G. Bontempi, P. Geurts, N. Harley, B. Lebichot, T. Lenaerts, and G. Louppe. Cham: Springer International Publishing, 2020, pp. 70–82.

[231] World Health Organization. *Global Health Expenditure Database*. URL: https://apps.who.int/nha/database (visited on 06/19/2023).

[232] World Health Organization. *Global spending on health: rising to the pandemic's challenges*. World Health Organization, 2022.

[233] J. Wu, Z. Huang, J. Thoma, D. Acharya, and L. Van Gool. "Wasserstein Divergence for GANs". In: *Proceedings of European Conference on Computer Vision (ECCV)*. 2018.

[234] C. Xiao, P. Zhong, and C. Zheng. "BourGAN: Generative Networks with Metric Embeddings". In: *Advances in Neural Information Processing Systems 32*. 2018.

[235] Z. Yang, M. Moczulski, M. Denil, N. De Freitas, A. Smola, L. Song, and Z. Wang. "Deep fried convnets". In: *IEEE International Conference on Computer Vision*. 2015, pp. 1476–1483.

[236] Z. Yin and Y. Shen. "On the dimensionality of word embedding". In: *Advances in neural information processing systems* 31 (2018).

[237] H. P. Young. "Monotonic solutions of cooperative games". In: *International Journal of Game Theory* 14.2 (June 1985), pp. 65–72. ISSN: 1432-1270. DOI: 10.1007/bf01769885. URL: http://dx.doi.org/10.1007/bf01769885.

[238] C. Zhang, H. Kjellstrom, and S. Mandt. "Determinantal point processes for mini-batch diversification". In: *Uncertainty in Artificial Intelligence* (2017).

[239] C. Zhang, C. Öztireli, S. Mandt, and G. Salvi. "Active mini-batch sampling using repulsive point processes". In: *AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 5741–5748.

[240] H. Zhang, Z. Zhang, A. Odena, and H. Lee. "Consistency regularization for generative adversarial networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2020.

[241] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. "Differentiable Augmentation for Data-Efficient GAN Training". In: *Neural Information Processing Systems* (2020).

[242] Y. Zhao, Z. Nasrullah, and Z. Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html.

[243] P. Zhong, Y. Mo, C. Xiao, P. Chen, and C. Zheng. "Rethinking generative mode coverage: A pointwise guaranteed approach". In: *Neural Information Processing Systems*. 2019, pp. 2088–2099.

[244] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. "FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 27268–27286.

Curriculum

Hannes De Meulemeester was born on November 25, 1991 in Oudenaarde, Belgium.

Degrees & Education

PhD student in Machine Learning, KU Leuven 2018–2023
PhD student, Department of Electrical Engineering, under the supervision of Prof. Bart De Moor.

MSc in Artificial Intelligence, KU Leuven 2016–2017

MSc in Information Engineering Technology, Ghent University 2012–2014

BSc in Applied Information Technology, University College Ghent 2009–2012

Science & Mathematics, Sint-Bernarduscollege, Oudenaarde 2003–2009

List of publications

Published Papers

- H. De Meulemeester and B. De Moor. "Unsupervised Embeddings for Categorical Variables". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207703

- H. De Meulemeester, J. Schreurs, M. Fanuel, B. De Moor, and J. A. Suykens. "The Bures metric for generative adversarial networks". In: *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*. Springer. 2021, pp. 52–66

- J. Schreurs, H. De Meulemeester, M. Fanuel, B. De Moor, and J. A. Suykens. "Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks". In: *Machine Learning, Optimization, and Data Science: 7th International Conference, LOD 2021, Grasmere, UK, October 4–8, 2021, Revised Selected Papers, Part II*. Springer. 2022, pp. 466–480

- A. Pandey, H. De Meulemeester, H. De Plaen, B. De Moor, and J. A. K. Suykens. "Recurrent Restricted Kernel Machines for Time-series Forecasting". In: *Proceedings of ESANN 2022* (2022)

- A. Pandey, H. De Meulemeester, B. De Moor, and J. A. Suykens. "Multi-view kernel PCA for time series forecasting". In: *Neurocomputing* 554 (2023), p. 126639. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.126639

Papers Under Review

- H. De Meulemeester, F. De Smet, J. van Dorst, E. Derroitte, and B. De Moor. “Explainable Unsupervised Anomaly Detection for Healthcare Insurance Data”. Internal Report 23-129, ESAT-STADIUS, KU Leuven (Leuven, Belgium), 2023.



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
STADIUS CENTER FOR DYNAMICAL SYSTEMS, SIGNAL PROCESSING AND DATA ANALYTICS
Kasteelpark Arenberg 10, bus 2446
B-3001 Leuven
hannes.demeulemeester@kuleuven.be
https://www.esat.kuleuven.be/stadius/