RIGA TECHNICAL UNIVERSITY

Faculty of Computer Science and Information Technology
Institute of Applied Computer Systems

Evalds Urtans
Doctoral Student of the Study Program Computer Systems

FUNCTION SHAPING IN DEEP LEARNING

The Doctoral Thesis

Scientific supervisor
professor Dr. habil. sc. ing.
Agris Nikitenko

RTU Press
Riga 2021

Abstract

The design of loss functions for deep learning methods is attracting growing attention because empirically found loss functions have achieved better results than commonly used loss functions that were derived from classical mathematical theory. This work describes the importance of loss functions and related methods for deep reinforcement learning and deep metric learning. A novel MDQN loss function outperformed DDQN loss function in PLE computer game environments, and a novel Exponential Triplet loss function outperformed the Triplet loss function with common additions in the face reidentification task with VGGFace2 dataset reaching 85.7% accuracy using zero-shot setting. This work also presents a novel UNet-RNN-Skip model to improve the performance of the value function for path planning tasks. It has the same policy outcome as the Value Iteration algorithm for 99.8% of the cases and can be trained on 32x32 maps, but then applied to larger maps like 256x256. Novel approaches have been usefully applied in multiple commercial applications for voice and face reidentification, audio signal denoising, and chromatography.

Contents

ABBREVIATIONS **4**

1 INTRODUCTION **5**
1.1 Importance of the subject 5
1.2 Objectives and thesis 7
1.3 Methodology 9
1.4 Scientific Novelty and Contributions of the Author 13
1.5 Structure of the Thesis 14

2 LITERATURE REVIEW **17**
2.1 Methodolgy of Literature Review 17
2.2 Results of Literature Review of Deep Metric Learning 19
2.3 Conclusions of Literature Review 30

3 IMPROVING THE PERFORMANCE OF FUNCTIONS USING DEEP LEARNING MODELS **31**
3.1 Value Iteration Algorithm 31
3.2 ConvNet and UNet models 33
3.3 RNN models 37
3.4 UNet-RNN-Skip model 40

4 FUNCTION SHAPING IN DEEP REINFORCEMENT LEARNING **44**
4.1 Q-Value and policy gradient functions for reinforcement learning 44
4.2 Deep Q-Network model and Loss function 45
4.3 Multi Deep Q-Network model and Loss function 48

5 FUNCTION SHAPING IN DEEP METRIC LEARNING **51**
5.1 Zero-Shot learning and Re-identification task 51
5.2 Triplet Loss Function 53
5.3 Exponential Triplet Loss Function 55

6 EXPERIMENTAL RESULTS AND APPLICATIONS **58**
6.1 Results of UNet-RNN-Skip model 58
6.2 Results of Multi Deep Q-Network Loss Function 62
6.3 Results of Exponential Triplet Loss Function 68
6.4 Practical applications 70

7 FUTURE RESEARCH **72**

8 CONCLUSIONS **73**

ACKNOWLEDGEMENTS **74**

BIBLOGRAPHY **75**

APPENDIX A - Paper 1 **87**

APPENDIX B - Paper 2 **94**

APPENDIX C - Paper 3 **137**

APPENDIX D - Paper 4 **153**

APPENDIX E - Paper 5 **161**

ABBREVIATIONS

A3C - Asynchronous Actor-Critic Agents
ACER - Actor-Critic with Experience Replay
AI - Artificial Intelligence
ASR - Automatic Speech Recognition
ConvNet, CNN - Convolutional Neural Network
CPU - Central Processing Unit
DQN - Deep Q-Network
DDPG - Deep Deterministic Policy Gradient
DDQN - Double Deep Q-Network
DNN - Deep Neural Network
DML - Deep Metric Learning
GPU - Graphical Processing Unit
GRU - Gated Recurrent Unit
HPC - High Performance Cluster
HPLC - High Performance Liquid Chromatography
IMAPLA - Importance Weighted Actor-Learner Architecture
LSTM - Long Short-Term Memory
MDQN - Multi Deep Q-Network
MERLIN - Memory, RL, and Inference Network
ML - Machine Learning
PLE - PyGame Learning Environment
PPO - Proximal Policy Optimization
RL - Reinforcement Learning
RNN - Recurrent Neural Network
ResNet - Residual Convolutional Neural Network
SLAM - Simultaneous Localization and Mapping
SLR - Systematic Literature Review
TD - Temporal Difference
UNet - U-Shaped Convolutional Neural Network
VI - Value Iteration Algorithm
VIN - Value Iteration Network

1 INTRODUCTION

1.1 Importance of the subject

In the past decade, Deep Machine Learning has taken over classical machine learning methods for approximating complex functions using high-dimensional datasets [115] [48] [118] [126]. Deep Machine Learning models are being used more frequently even to extract functions that describe patterns in the datasets or processes they are observing in an unsupervised or semi-supervised manner [69] [83] [19] [22]. These kinds of models are trained using loss functions to extract deep representations that provide insight into the underlying manifold of features, patterns, and logic. Nowadays, loss functions and model architecture itself have become research subjects instead of feature engineering and rule-based pattern recognition of data inputs [57]. Deep Machine Learning models achieve the highest accuracy on image classification tasks [107] [45], natural language modeling tasks [10] [95], automated speech recognition tasks [78] [85], time-series tasks [9] [72] and other tasks where inputs or outputs have high dimensionality. Deep Machine Learning models also achieve the state of the art performance in Reinforcement Learning tasks in computer game environments and robotics, where only a human operator previously was capable of producing inputs [35] [103] [21] [76]. Deep Machine Learning methods also provide comparable transparency of reasoning to classical statistical methods [63] [93].

The goal of the thesis is to develop novel loss functions and models based on empirical research. These loss functions and models should achieve the higher performance on selected datasets than existing loss functions and models.

Historically, most loss functions are derived from statistics, probability, and information theory, but recently, empirically discovered loss functions in some tasks have shown superior results [61] [89].

To achieve the goal, the following topics have been explored:

1. The construction of novel loss functions for Deep Reinforcement Learning and Deep Metric Learning. Loss functions in reinforcement Learning has been evaluated on computer game environments to achieve higher score. Similarly, loss functions in Deep Metric Learning has been evaluated on face re-identification task and voice re-identification task.

2. The construction of model architecture and training procedure for approximating the Value Function and improving the performance of the Value Iteration algorithm.

3. Experimental evaluation of novel methods using synthetic, academic, and private datasets and environments to probe their usability in practical applications.

The novel loss functions developed in the thesis for Deep Metric Learning tasks have achieved 85.7% accuracy in the face reidentification task, whereas the standard loss function achieved 78.6% accuracy. The accuracy has been achieved in a zero-shot learning setting [23] [116]. The zero-shot learning for the face reidentification task ensures that the face of a person has not been seen during the training process, but it is only matched to the most similar face between enrollment and reidentification samples afterwards during the inference phase. The same loss function in the zero-shot setting has been used for voice reidentification tasks and achieved an accuracy of 88.4% on private datasets. Similarly, the novel loss functions developed in the thesis for the field of Deep Reinforcement learning achieved higher scores in PyGame Learning Environment. These novel loss functions have also been successfully applied in a private research of solvent gradient optimization for compound separation in analytical chemistry. Finally, novel Deep Learning models have been developed in the thesis to improve the performance of the Value Function in pathfinding tasks for mobile robots. The novel model is capable of approximating the Value Function and can be executed in parallel. It produces a value map by an order faster than the standard Value Iteration Algorithm.

1.2 Objectives and thesis

The thesis improves the deep machine learning training process performance and results in practical applications in the tasks of DML and RL by introducing novel loss functions and model architectures. Novel loss functions of this research should converge faster during the training, achieve better results, and should be usable in different tasks starting from classification for face reidentification, analytical chemistry, and reinforcement learning for controlling agents in complex environments. Convergence in the context of Deep Learning is achieved when the following equation is true $|\frac{\mathcal{L}(f_\theta(x),y)_i}{\mathcal{L}(f_\theta(x),y)_{i-1}}| < \delta$ where the relative difference between average $\mathcal{L}$ loss function output regarding inputs x and ground truth y in a current epoch and previous epoch is smaller than δ. Depending on the cleanness of the data and the type of task, δ could be between 0.1-0.001. Outputs of a loss function regarding the dataset or environment must decrease and converge. Usually loss functions have a global minimum of a value of zero, except for reinforcement learning where it might not be possible to estimate the ground truth that would give the highest reward.

The objectives of the thesis document are following:

1. Perform review of existing loss functions for functions similar to Deep Q-Learning [68] and Triplet Loss [23].
2. Develop novel loss functions similar to Deep Q-Learning and Triplet Loss using zero-shot learning.
3. Develop a novel model to learn approximate the Value function in Value Iteration algorithm [88].
4. Evaluate results of a novel Deep Q-Learning loss functions in game environments and in applications of analytical chemistry.
5. Evaluate results of a novel Triplet Loss functions in the face reidentification task.
6. Evaluate results of a novel model of approximation of Value Function and compare it to the full Value Iteration algorithm.
7. Publish findings in scientific publications and include in this thesis.

The list of theses of this paper are following:

1. Novel MDQN loss function in the tasks of Deep Q-Learning outperforms DQN loss functions.

2. Novel Exponential Triplet Loss function in the tasks of Deep Metric Learning to outperform Triplet Loss function.

3. Novel UNet-RNN-Skip model improves the performance of Value Function in the context of Value Iteration algorithm.

4. Novel MDQN and Triplet Loss functions can be used in practical applications for face re-identification, voice re-identification, noise removal for speech and chromatography in analytical chemistry.

1.3 Methodology

Objectives of the thesis are accomplished by analytical and experimental research that has been published in the scientific research papers listed in 1.5. sec.

In the thesis, experimental and data analysis research methods have been used.

Qualitative and quantitative research methods have been used to review the scientific literature, existing and novel methods.

Within this research, the primary research subject has been novel loss functions and methods to improve performance and results for DML (Deep Metric Learning) and RL (Reinforcement Learning).

The process of the novel loss function design is shown in 1. fig.

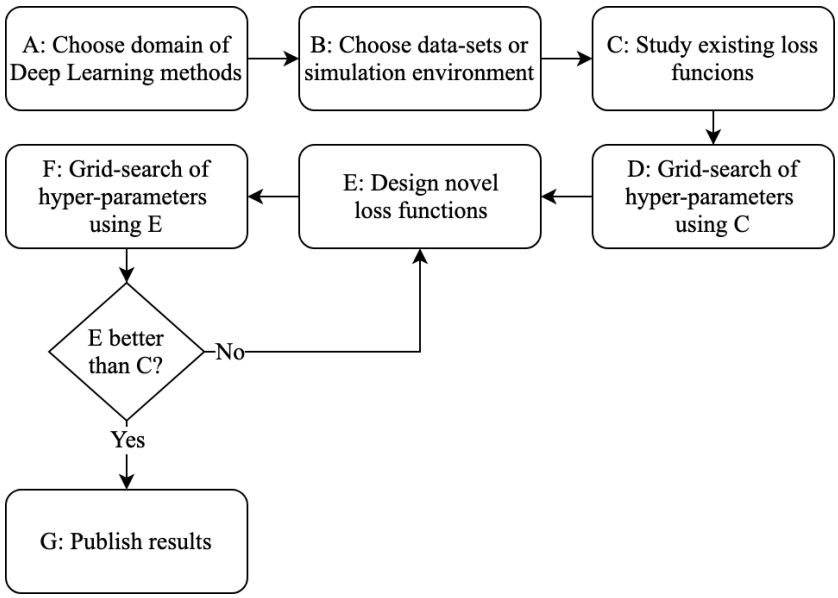

Figure 1: Simplified activity diagram of the process of novel loss function design in the context of Deep Learning.

The process starts by choosing a problem space (A) suitable for the Deep Learning. In this research, the problem spaces include: DML based embedding for face and voice reidentification tasks; Deep RL for Q-Value based policy learning in environments of computer games; analytical chemistry for

finding solvent gradients that separate the peaks of compounds using Deep RL.

Next, it is necessary to look for suitable datasets or testing environments (B). In this research, face reidentification and image classification datasets have been used as well as computer game environments for testing agents in the context of RL.

Then the existing loss functions (C) have been analyzed from the scientific literature and the latest publications. These functions then have been implemented in Deep Learning frameworks like PyTorch [79] and their characteristics w.r.t. (with regard to) Input parameters have been studied.

Then the existing loss functions have been tested on the chosen dataset or environment by searching for the best hyper-parameter combinations (D). Hyper-parameters are parameters of a loss function or model that are not learnable during the training process, but are predefined in the beginning of the process. Each training run is executed until it reaches the convergence of the error w.r.t. train and loss functions. The convergence in Deep Learning training is achieved when the output of the loss function w.r.t. train dataset in between training epochs does not change significantly (for example, under 0.1%) and at the same time the output of the loss function w.r.t. validation dataset does not increase. If the train and test functions diverge in further epochs, then it might indicate overfitting. Epoch in Deep Learning is a complete iteration of the whole training dataset and parameter fitting of a model w.r.t. loss function. Grid-search of hyper-parameters must be done for every loss function because the same hyper-parameters often are not optimal for every specific loss function and the comparison would be biased without such a search process.

Next, a creative process of design of a novel loss function occurs that takes into account the mathematical theory and design of previous loss functions. It involves analyzing the shapes and desirable properties of a loss function w.r.t. input parameters and the gradient of the error. This process could be done also by automatic function construction algorithms and optimization methods such as Bayesian Optimization, Monte-Carlo Optimization methods, or Genetic Algorithms. However, in practice, such loss function design often is impractical, because the grid search of hyper-parameters for each loss function alone takes a long time, but for automatic development of loss functions hyper-parameters in the process itself also will need a grid search as well. Hyper-parameter searches were done on HPC (High-Performance Cluster), it took weeks or even months depending on the dataset to search

hyper-parameter combinations for a single change in a loss function, training algorithm, and in a model architecture.

Then, as described, a full hyper-parameter search (F) is done on a novel loss function in the same manner as for the existing loss functions.

If a novel loss function yielded better results than the existing loss function, then the results were published in the scientific literature (G), but if a novel loss function were inferior, then it was necessary to return to the design phase of the loss function (E).

For the research of each of the novel loss functions, the following methodology has been applied:

1. Review of similar loss functions, models and training procedures.

2. Implementation and testing of multiple candidates of novel loss functions, models and training procedures.

3. Testing novel and existing loss functions on benchmark data-sets or environments in case of reinforcement learning. For each of data-sets or environments grid-search of hyper-parameters has been done for a fair comparison.

4. Ablation studies by comparing individual parts of loss functions and training procedures.

For Deep Q-Learning loss functions, a survey of all state of the art methods at the time of publication has been done [109]. Deep Q-Learning loss functions have been tested in at least 4 PyGame Learning Environment games [104] and have been tested in the field of chromatography of analytical chemistry [24].

Later, a novel Exponential Triplet Loss function [110] have been developed and tested on different datasets for zero-shot reidentification tasks like VGGFace2 [13], EMNIST [15] and CIFAR10 [47]. Data-sets have been reordered so that the class samples included in the test dataset would not be included also in the training data-set to ensure zero-shot compatible models [71].

Finally, VI (Value Iteration) function has been modelled with UNet-RNN-Skip and compared with different variants of UNet [87] models to improve the speed of VI algorithm using deep learning methods. A novel synthetic dataset generator has been created to validate VI and can be used also to validate other policy models.

Implementations are made publicly available as open-source repositories.

They have been developed using PyTorch framework [79]. PyTorch has been chosen for its capabilities of creating a dynamic function graph and easy debugging and overriding of function gradients.

For each set of methods and loss functions, a full grid search of hyper-parameters has been executed using RTU HPC (High Performance Cluster) that provides access to Nvidia GPUs V100 and K40.

1.4 Scientific Novelty and Contributions of the Author

The Thesis includes: a novel loss function MDQN for Deep Reinforcement learning described in detail in 4.3. sec., a novel loss function, Exponential Triplet loss for Deep Metric learning is described in detail in 5.3. sec., a novel embedding space normalization functions Unit-Range and Unit-Bounce a novel model UNet-RNN-Skip for improving the performance of the Value function for policy selection task for 2D representations of environments are described in detail in 3.4. sec., a novel synthetic dataset generator OccupancyMapGenerator for 2D mapping tasks.

The author is listed as the main author of all publications included in the appendix, except a publication in High Performance Liquid Chromatography where the author was the main author regarding Deep Learning approaches using Deep Reinforcement learning and other co-authors did research in the chemistry field.

The research work included in the thesis has been published in 3 scientific conferences and 1 scientific monograph. The works have won multiple awards:

1. Best research paper in ICCDA 2020, USA
2. 3rd best doctorate research in all sciences, ResearchSlam 2018, Latvia

1.5 Structure of the Thesis

The thesis consists of 165 pages. It is divided into eight main sections. The thesis is written in the form of a collection of publications with extended explanations as due to the limitations of conference papers, it was not possible to include the whole background of the research in the publications themselves.

The structure of the thesis are described below:

1. sec. Introduction, research background, research motivation, and research objectives.

2. sec. Contains a literature review of Deep Metric Learning loss functions to give better background information on the topic of loss function design. Similar survey of loss functions for Q-Value functions has been done also in one of the attached publications [109].

2. sec. Gives background of the research done in the design of novel loss functions that are being used in Deep Metric Learning (DML). It highlights active research areas, methods, and practical applications.

3. sec. Describes the problem of the Value Iteration Algorithm, gives an introduction to the theoretical background of ConvNet, UNet, and RNN, and describes the novel UNet-RNN-Skip model. UNet-RNN-Skip has been trained to mimic the Value function and reduce the number of iterations required to converge the optimal policy. Novel model is used to learn another existing non-paralellizable function and make it parallelizable and thus improve its convergence speed.

4. sec. Describes Q-Value function-based Deep Reinforcement Learning methods, test environments and approaches for validating results, and a novel MDQN loss function that has been tested and analyzed within PyGame Learning environment and in liquid chromatography method optimization tasks.

5. sec. Describes Deep Metric Learning to use zero-shot embedding models that have been trained using triplet loss and a novel Exponential Triplet loss function. It has been tested on multiple datasets in the context of the sample reidentification task.

6. sec. Describes the experimental results of a novel model and loss function described in the previous sections as well as describes practical applications where these models and loss functions have been already applied successfully.

7. sec. Describes future research topics discovered by studying novel models and loss functions described in the thesis.

8. sec. Summarizes the main contributions of this study.

Appendix Contains a list of publications that describe and research directions of the thesis, theoretical and experimental results.

The link between objectives, topics, and publications has been illustrated in 2. fig.

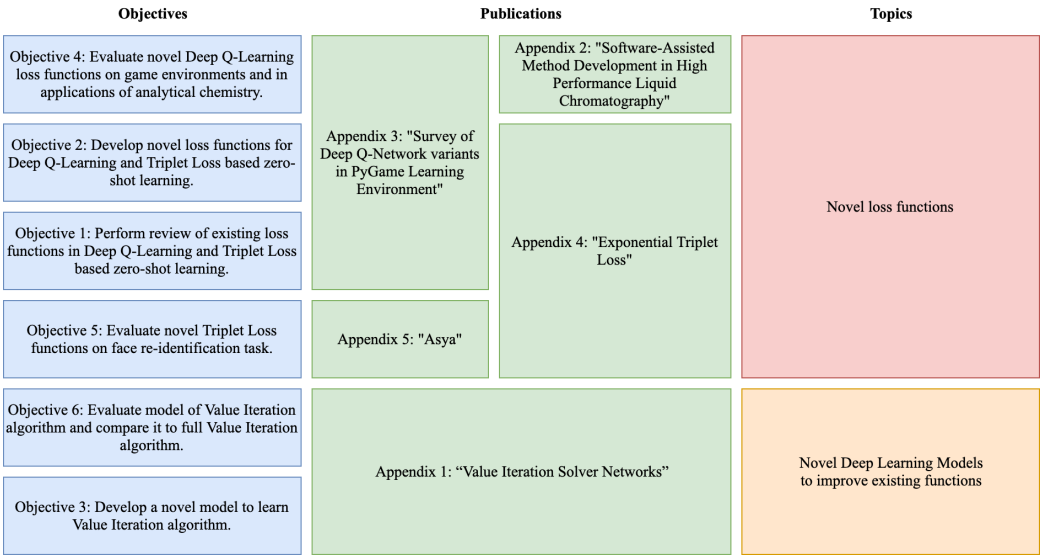

Figure 2: Interconnection of published papers, thesis objectives, and research topics.

The list of publications included in the thesis and the main contributions are described below:

Appendix A **Value Iteration Solver Networks, International Conference on Intelligent Autonomous Systems, 2020, IEEE, Evalds Urtans, Valters Vecins.** Introduced a novel model UNet-RNN-Skip for improving the performance of the Value Iteration Algorithm. Introduced also a novel synthetic dataset generator OccupancyMapGenerator for evaluation of path planning models.

Appendix B **Software-Assisted Method Development in High Performance Liquid Chromatography"; ISBN: 978-1-78634-545-5, Sep. 2018, Sergey V. Galushko, Irina Shishkina, Evalds Urtans, Oksana Rotkaja.** Introduced a novel Deep Reinforcement Learning based method for sequentially developing solvent gradients in HPLC.

Appendix C **Survey of Deep Q-Network variants in PyGame Learning Environment, International Conference on Deep Learning technologies, 2018, ACM, Evalds Urtans, Agris Nikitenko** Introduced a novel Deep Reinforcement Learning based method and a novel MDQN loss function.

Appendix D **Exponential Triplet Loss, International Conference on Compute and Data Analysis, 2020, IEEE/ACM, Evalds Urtans, Valters Vecins, Agris Nikitenko** Introduced a novel Deep Metric Learning based loss function Exponential Triplet Loss.

Appendix E **"asya: Mindful verbal communication using deep learning", Cornell University, Computing Research Repository, 2020, Evalds Urtans, Austris Tabaks** Introduced a novel system based on Exponential Triplet Loss for voice reidentification.

2 LITERATURE REVIEW

This section aims to provide a background of existing research in the loss functions of the Deep Metric Learning. It explains the importance of the research and activity in the development of novel loss functions. Similar survey of methods and loss functions has been done also for Deep Reinforcement Learning [109]. Results of the findings for Deep Reinforcement Learning have been included in 8. sec.

2.1 Methodolgy of Literature Review

The methodology of SLR (Systematic Literature Review) presented in this document is based on a systematic mapping study [80] [43]. The results of SLR contain the map of clusters based on the origins of loss functions and methods, as well as a qualitative review based on research questions. The results also include a list of limitations identified for loss functions and methods used in the reviewed papers.

The method for selecting and evaluating papers contains the following steps listed in 3. fig. Initially the most well-known publications [7] [23] in the field of deep metric learning (DML) have been selected. Additionally, the following keywords were used for the initial search of papers: triplet loss, contrastive loss, ranking loss, deep metric learning, representation learning, one-shot learning, zero-shot learning, product reidentification task, signature reidentification, face reidentification task. Then the publications have been thoroughly analyzed and documented to check if publications match the field of DML loss function research. Then matching to Quality Assessment criteria has been evaluated. Then, if at least single assessment criteria have been met, a publication has been added to the main list. In addition, if answers to research questions have been found in selected publications, then those have been documented. Then the references and citations of this publication have been found. For each of the relevant publications, their citation count has been found and divided by years passed since publishing. Then those with the highest value of influence would be analyzed first.

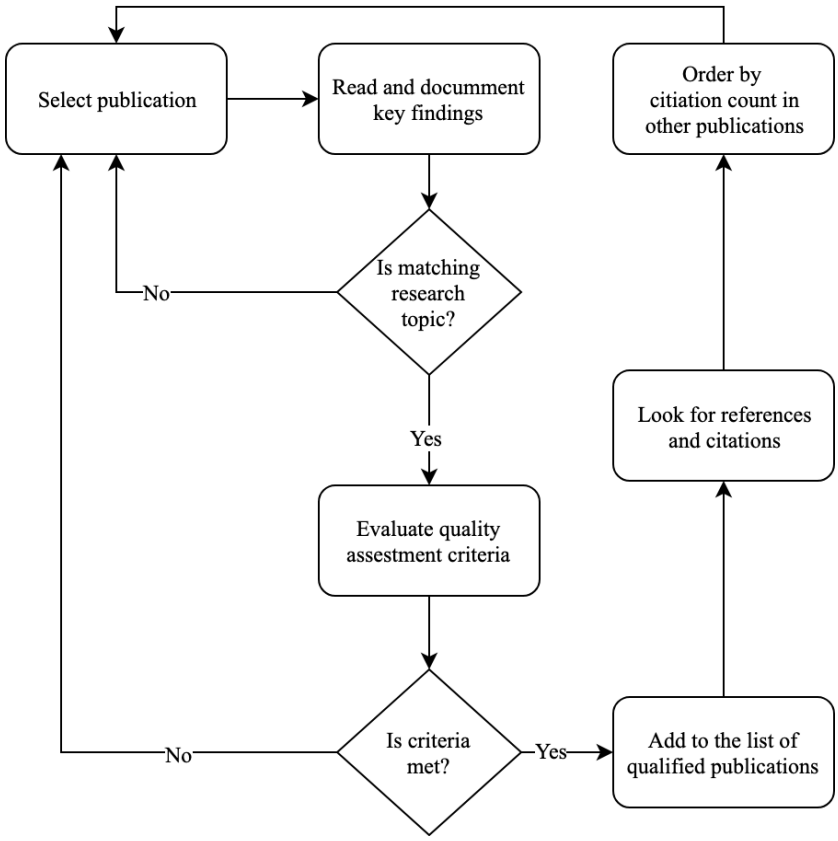

Figure 3: The methodology of SLR.

To find a valid direction of further research, few research questions (RQ) were selected. The research questions addressed by this study are:

- RQ1: What kinds of functions have been studied similar to Triplet Loss functions?
- RQ2: Do the novel loss functions achieve significantly better results than previous functions?
- RQ3: Do the novel loss functions have theoretical grounding, or are they purely empirical?
- RQ4: What are the limitations of novel loss functions?

2.2 Results of Literature Review of Deep Metric Learning

The results of SLR regarding DML are mapped in multiple tables depending on the relevant properties extracted from papers. In 1. tab. information about authors, affiliation, country of origin and conferences regarding DML have been listed. Publications have been ordered by the year of publishing, and the numbering of publications has been maintained also in the following tables.

Table 1:

Authors and conferences of studies regarding DML.

No	Title	Authors	Affiliation	Country	Year	Conference / Journal
1	Signature Verification Using A "Siamese" Time Delay Neural Network [7]	J. Bromley, J. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Sckinger, R. Shah	AT&T Bell laboratories	USA	1993	INT J PATTERN RECOGN
2	Neighbourhood Components Analysis [27]	J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov	AT&T Bell laboratories	Canada	2004	NIPS
3	Learning a Similarity Metric Discriminatively, with Application to Face Verification [14]	S. Chopra, R. Hadsell, Y. LeCun	NYU	USA	2005	CVPR
4	Distance metric learning for large margin nearest neighbor classification [127]	K. Q. Weinberger, L. Saul	Yahoo!, University of California	USA	2005	NIPS
5	Large scale metric learning from equivalence constraints [46]	M. KŽstinger, M. Hirzer, P. Wohlhart, P. Roth, H. Bischof	Graz University of Technology	Austria	2012	CVPR
6	Quadruplet-Wise Image Similarity Learning [52]	M. Law, N. Thome, M. Cord	Sorbonne University	France	2013	ICCV
7	Reidentification by Relative Distance Comparison [135]	W. Zheng, S. Gong, T. Xiang	College of Electronic and Information, South China University of Technology	China	2013	TPAMI
8	Deep Metric Learning for Practical Person Re-Identification [132]	D. Yi, Z. Lei, S. Li	IEEE	China	2014	ArXiv
9	FaceNet: A unified embedding for face recognition and clustering [23]	F. Schroff, D. Kalenichenko, J. Philbin	Google	USA	2015	CVPR
10	Improved Deep Metric Learning with Multi-class N-pair Loss Objective [98]	K. Sohn	NEC	USA	2016	NIPS
11	A Discriminative Feature Learning Approach for Deep Face Recognition [128]	Y. Wen, K. Zhang, Z. Li, Y. Qiao	SIAT	China	2016	ECCV

12	Deep Metric Learning via Lifted Structured Feature Embedding [100]	H. O. Song, Y. Xiang, S. Jegelka, S. Savarese	Stanford University, MIT	USA	2016	CVPR
13	Deep clustering: Discriminative embeddings for segmentation and separation [34]	J. Hershey, Z. Chen, J. Le Roux, S. Watanabe	Mitsubishi, Columbia University	USA	2016	ICASSP
14	Learning Deep Embeddings with Histogram Loss [113]	E. Ustinova, V. Lempitsky	Skoltech	Russia	2016	NIPS
15	Local Similarity-Aware Deep Feature Embedding [37]	C. Huang, C. C. Loy, X. Tang	The Chinese University of Hong Kong, SenseTime Group Limited	China	2016	NIPS
16	Metric Learning with Adaptive Density Discrimination [86]	O. Rippel, M. Paluri, P. DollĞr, L. D. Bourdev	Facebook	USA	2016	ICLR
17	L2-constrained Softmax Loss for Discriminative Face Verification [84]	R. Ranjan, C. D. Castillo, R. Chellappa	UMIACS	USA	2017	ArXiv
18	In Defense of the Triplet Loss for Person Re-Identification [33]	A. Hermans, L. Beyer, B. Leibe	RWTH	Germany	2017	ArXiv
19	Deep Metric Learning with Angular Loss [117]	J. Wang, F. Zhou, S. Wen, X. Liu, Y. Lin	Baidu	China	2017	ICCV
20	No Fuss Distance Metric Learning Using Proxies [70]	Y. Movshovitz-Attias, A. Toshev, T. Leung, S. Ioffe, S. Singh	Google	USA	2017	ICCV
21	Sampling Matters in Deep Embedding Learning [64]	R. Manmatha, C. Y. Wu, A. Smola, P. KrŁhenb§hl	UT Austin, Amazon	USA	2017	ICCV
22	Deep Metric Learning via Facility Location [99]	H. O. Song, S. Jegelka, V. Rathod, K. Murphy	Google	USA	2017	CVPR
23	Deep spectral clustering learning [53]	M. Law, R. Urtasun, R. Zemel	University of Toronto	Canada	2017	ICML
24	Hard-Aware Deeply Cascaded Embedding [133]	Y. Yuan, K. Yang, C. Zhang	MOE, Peking University, DeepMotion, Microsoft Research	China	2017	ICCV
25	PPFNet: Global Context Aware Local Features for Robust 3D Point Matching [17]	H. Deng, T. Birdal, S. Ilic	TMU, NUDT	Germany, China	2018	CVPR
26	Ranked List Loss for Deep Metric Learning [121]	X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, N. Robertson	Anyvision, QueenŌs University Belfast	UK	2019	CVPR
27	Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning [119]	X. Wang, Xintong Han, W. Huang, D. Dong, M. Scott	Malong Technologies	China	2019	CVPR
28	A Simple and Effective Framework for Pairwise Deep Metric Learning [82]	Q. Qi, Y. Yan, Z. Wu, X. Wang, T. Yang	The Chinese University of Hong Kong	China	2019	ECCV

29	Deep Metric Learning Meets Deep Clustering: An Novel Unsupervised Approach for Feature Embedding [73]	B. X. Nguyen, B. D. Nguyen, G. Carneiro, E. Tjiputra, Q. D. Tran, T. T. Do	AIOZ	Singapore	2020	ArXiv
30	Exponential triplet loss [110]	E. Urtans, A. Nikitenko, V. Vecins	RTU	Latvia	2020	ICCDA

In, 3. tab. information about novel loss functions and their properties regarding DML have been listed. Embedding space refers to normalization or measurement methods between two or more vectors in a latent space. Each of the embedding vectors has been produced by a deep learning based model for the data point. Then two or more embedding vectors have been processed using the loss function and a deep learning based model weights are calculated using the back-propagation algorithm. In addition, for many of these papers uses sample mining methods to select the best training samples to improve the results and speed of the training.

Table 3:

Novel loss functions of studies regarding DML.

No	**Title**	**Year**	**Embedding space**	**Sample Mining**	**Loss function**
1	Signature Verification Using A "Siamese" Time Delay Neural Network [7]	1993	Euclidean	None	Contrastive loss
2	Neighbourhood Components Analysis [27]	2004	Euclidean, Mahalanobis	None	NCA Loss
3	Learning a Similarity Metric Discriminatively, with Application to Face Verification [14]	2005	L1, Euclidean	None	Contrastive Loss
4	Distance metric learning for large margin nearest neighbor classification [127]	2005	Euclidean, Mahalanobis	None	Triplet Hinge Loss
5	Large scale metric learning from equivalence constraints [46]	2012	Mahalanobis	None	KISS-BCE Loss
6	Quadruplet-Wise Image Similarity Learning [52]	2013	Qwise	None	Quadruplet Hinge Loss
7	Reidentification by Relative Distance Comparison [135]	2013	RDC	None	RDC Loss
8	Deep Metric Learning for Practical Person Re-Identification [132]	2014	Cosine distance	Hard	Binomial Deviance Loss
9	FaceNet: A unified embedding for face recognition and clustering [23]	2015	L2, Euclidean	Hard, Semi-Hard	Triplet Loss, Harmonic Triplet Loss
10	Improved Deep Metric Learning with Multi-class N-pair Loss Objective [98]	2016	L2, Cosine distance	N Hard Mining	multi-class N-pair loss
11	A Discriminative Feature Learning Approach for Deep Face Recognition [128]	2016	Cosine distance	None	Center loss

12	Deep Metric Learning via Lifted Structured Feature Embedding [100]	2016	L2, Euclidean	Mining positives	Lifted Structured Loss, Lifted Struct
13	Deep clustering: Discriminative embeddings for segmentation and separation [34]	2016	L2, Euclidean	None	Pairwise metric Loss
14	Learning Deep Embeddings with Histogram Loss [113]	2016	Cosine distance	None	Histogram Loss
15	Local Similarity-Aware Deep Feature Embedding [37]	2016	PDDM	Hard mining	PDDM - Double Header Hinge Loss
16	Metric Learning with Adaptive Density Discrimination [86]	2016	Euclidean	Neighbourhood Sampling	Magnet Loss
17	L2-constrained Softmax Loss for Discriminative Face Verification [84]	2017	Cosine distance	None	L2 constrained Softmax Loss
18	In Defense of the Triplet Loss for Person Re-Identification [33]	2017	L2, Euclidean	None	Batch All Triplet Loss
19	Deep Metric Learning with Angular Loss [117]	2017	Angle	None	Angular loss
20	No Fuss Distance Metric Learning Using Proxies [70]	2017	L2, Euclidean	None	Proxy Ranking Loss, Proxy NCA Loss
21	Sampling Matters in Deep Embedding Learning [64]	2017	L2, Euclidean	Distance weighted sampling	Triplet Loss, Contrasitve Loss
22	Deep Metric Learning via Facility Location [99]	2017	L2, Euclidean	None	Struct Clust, Clustering Loss
23	Deep spectral clustering learning [53]	2017	L2, Euclidean	None	Spectral Clustering Loss
24	Hard-Aware Deeply Cascaded Embedding [133]	2017	Euclidean	Model-based	Any / Contrastive loss
25	PPFNet: Global Context Aware Local Features for Robust 3D Point Matching [17]	2018	L2, Euclidean	None	N-Tuple loss
26	Ranked List Loss for Deep Metric Learning [121]	2019	Euclidean	Hard	Ranked List Loss
27	Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning [119]	2019	Cosine distance	Hard	Multi-Similarity Loss
28	A Simple and Effective Framework for Pairwise Deep Metric Learning [82]	2019	Euclidean	TopK Loss mining	DRO-TopK Loss
29	Deep Metric Learning Meets Deep Clustering: An Novel Unsupervised Approach for Feature Embedding [73]	2020	L2, Euclidean	None	Unsupervised UDML Loss
30	Exponential triplet loss [110]	2020	Unit-Range	Hard	Exponential Triplet Loss

In 4. fig. relationship of DML Loss functions has been summarized. Colors denote similar groups of loss functions by their origin and methodology. It is possible to observe that most of the loss functions come from the seminal works of Contrastive Loss [7] and NCA Loss [27] and Triplet Loss [127]. Most of the functions are extensions of simple Hinge Loss [127]. As seen in 3. tab. most of the loss functions use sample mining methods, because they are trained only using a few data samples per training iteration. Some methods like Histogram Loss [113] or Quadruplet Hinge Loss [52] use more samples per training iteration, but their results on benchmark datasets are not significantly better than other methods as seen in 5. tab.

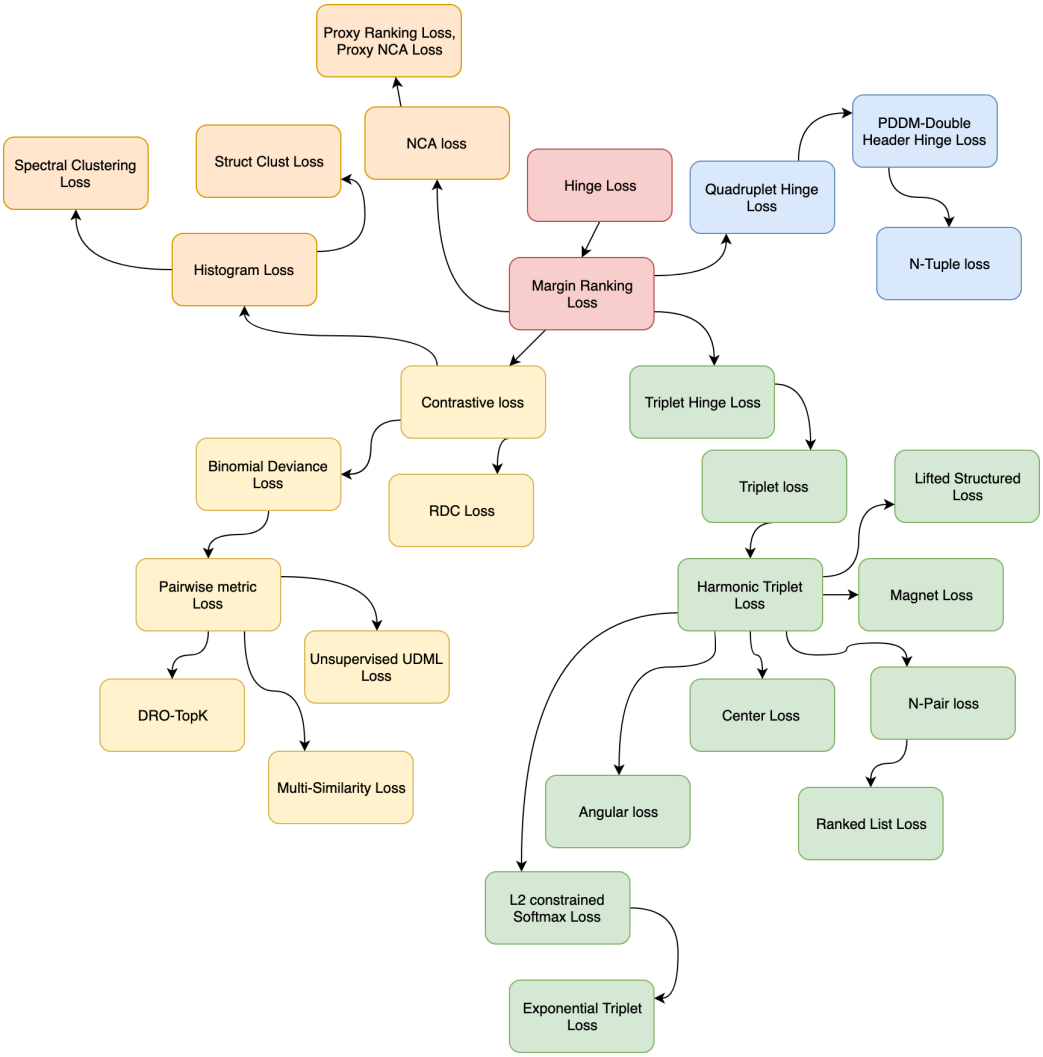

Figure 4: Relationship of DML Loss functions. Colors denote similar groups of loss functions by their origin and methodology.

5. tab. lists practical applications for each of DML loss functions that have been studied, as well as their benchmark datasets and the best results on those datasets. Where applicable, Top-1 accuracy has been selected for the best results on each of the datasets. As seen in the listings, most of the practical applications and datasets have been used for face and product re-identification.

Table 5:

Practical applications and best results for every dataset.

No	Title	Year	Practical application	Dataset / Top-1 Acc.
1	Signature Verification Using A "Siamese" Time Delay Neural Network [7]	1993	Signature re-identification	Signatures: 97%
2	Neighbourhood Components Analysis [27]	2004	Handwriting identification, Face re-identification	USPS: 85% FERET-B
3	Learning a Similarity Metric Discriminatively, with Application to Face Verification [14]	2005	Face Re-identification	AT&T: 92.5%
4	Distance metric learning for large margin nearest neighbor classification [127]	2005	Handwriting identification, text classification	MNIST: 98.8% Letters: 96.3% 20news: 92% Isolet: 96.6% YaleFaces: 93.9%
5	Large scale metric learning from equivalence constraints [46]	2012	Face Re-identification, Image Re-idenification	LFW: 80.5% VIPeR: 22%
6	Quadruplet-Wise Image Similarity Learning [52]	2013	Product or image retrieval	OSR: 74.6% Pubfig: 77.6%
7	Reidentification by Relative Distance Comparison [135]	2013	Face Re-identification	ETHZ: 61.58% i-LIDS: 32.60% VIPeR: 9.12%
8	Deep Metric Learning for Practical Person Re-Identification [132]	2014	Face Re-identification	VIPER: 34.49%
9	FaceNet: A unified embedding for face recognition and clustering [23]	2015	Face Re-identification	LFW: 99.63% YTF: 95.12%
10	Improved Deep Metric Learning with Multi-class N-pair Loss Objective [98]	2016	Product image retrieval, Face Re-identification	LFW: 98.33% SOP: 28.19% CAR-196: 33.5% CUB-200: 27.24%
11	A Discriminative Feature Learning Approach for Deep Face Recognition [128]	2016	Face Re-identification	LFW: 99.28% YTF: 94.9% MegaFace: 76.5%
12	Deep Metric Learning via Lifted Structured Feature Embedding [100]	2016	Product or image retrieval	CUB200: 55%, CARS196: 48%, SOP: 62%
13	Deep clustering: Discriminative embeddings for segmentation and separation [34]	2016	Speaker diarization, seperation	WSJ0: 2.74 dB (SDR)
14	Learning Deep Embeddings with Histogram Loss [113]	2016	Product or image retrieval	CUHK03: 65.7% CUB-200: 51% Market-1501: 59.47% SOP: 65%
15	Local Similarity-Aware Deep Feature Embedding [37]	2016	Product or image retrieval	CARS196: 57.4% CUB-200: 58.3% ImageNet: 48.2%
16	Metric Learning with Adaptive Density Discrimination [86]	2016	Image classification, Face Re-identification	Stanford Dogs: 75.1% Flowers-102: 91.4% Oxford-IIIT Pet: 89.4% ImageNet: 84.1%
17	L2-constrained Softmax Loss for Discriminative Face Verification [84]	2017	Image classification, Face Re-identification	LFW: 99.33% YTF: 99.78% MNIST: 99.05% IJB-A: 97.5%
18	In Defense of the Triplet Loss for Person Re-Identification [33]	2017	Product image retrieval, Face Re-identification	MARS: 90.53%, Market-1501: 79.8%, CUHK03: 87.58%

19	Deep Metric Learning with Angular Loss [117]	2017	Product or image retrieval	CAR-196: 71.4%, CUB-200: 54.7%, SOP: 70.9%
20	No Fuss Distance Metric Learning Using Proxies [70]	2017	Product or image retrieval	CARS196: 73:22% CUB200: 73.22% SOP: 73.73%
21	Sampling Matters in Deep Embedding Learning [64]	2017	Product or image retrieval, Face Re-identification	CARS196: 86.9% CUB200: 63.9% SOP: 72.7%
22	Deep Metric Learning via Facility Location [99]	2017	Product or image retrieval	CARS196: 58.11% CUB200: 48.18% SOP: 67.02%
23	Deep spectral clustering learning [53]	2017	Product or image retrieval	CARS196: 73.07% CUB200: 43.22% SOP: 67.59%
24	Hard-Aware Deeply Cascaded Embedding [133]	2017	Product or image retrieval	CARS196: 83.8% CUB-200: 60.7% In-shop: 62.1 % SOP: 70.1%
25	PPFNet: Global Context Aware Local Features for Robust 3D Point Matching [17]	2018	3D Point Cloud matching	SUN3D: 71%
26	Ranked List Loss for Deep Metric Learning [121]	2019	Product or image retrieval	CARS196: 82.1% CUB-200: 61.3% SOP: 79.8%
27	Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning [119]	2019	Product or image retrieval	CARS196: 77.3% CUB-200: 65.7% In-Shop: 78.2%
28	A Simple and Effective Framework for Pairwise Deep Metric Learning [82]	2019	Product or image retrieval	In-shop: 91.3% CARS-196: 86.2% CUB-200: 68.1%
29	Deep Metric Learning Meets Deep Clustering: An Novel Unsupervised Approach for Feature Embedding [73]	2020	Product or image retrieval	CUB200: 47.5%, Car196: 42.6%
30	Exponential triplet loss [110]	2020	Face Re-identification, Image Re-idenification	VGGFace2: 85.7% EMNIST: 86% FMNIST: 93.1% CIFAR10: 87.3% MNIST: 99.6%

The Quality Assessment (QA) criteria are listed below:

- QA1: Does the publication provide open-source implementation of a novel loss function or methodology?
- QA2: Does a publication have achieved state-of-the-art results on the datasets it studies?
- QA3: Does the publication provide a theoretical proof of a novel loss function or methodology?
- QA4: Does the publication include an ablation study to test effects on the results of functional parts one by one?
- QA5: Does a publication have over 100 citations?

Table 7:

Evaluation of quality of publications baset on criteria.

No	Title	QA1	QA2	QA3	QA4	QA5	Total
11	A Discriminative Feature Learning Approach for Deep Face Recognition [128]	Yes	Yes	No	Yes	Yes	4
18	In Defense of the Triplet Loss for Person Re-Identification [33]	Yes	Yes	No	Yes	Yes	4
21	Sampling Matters in Deep Embedding Learning [64]	No	Yes	Yes	Yes	Yes	4
23	Deep spectral clustering learning [53]	No	Yes	Yes	Yes	Yes	4
28	A Simple and Effective Framework for Pairwise Deep Metric Learning [82]	Yes	Yes	Yes	Yes	No	4
3	Learning a Similarity Metric Discriminatively, with Application to Face Verification [14]	No	Yes	Yes	No	Yes	3
5	Large scale metric learning from equivalence constraints [46]	No	Yes	Yes	No	Yes	3
9	FaceNet: A unified embedding for face recognition and clustering [23]	No	Yes	No	Yes	Yes	3
16	Metric Learning with Adaptive Density Discrimination [86]	Yes	Yes	No	No	Yes	3
17	L2-constrained Softmax Loss for Discriminative Face Verification [84]	No	Yes	No	Yes	Yes	3
19	Deep Metric Learning with Angular Loss [117]	No	Yes	Yes	No	Yes	3
20	No Fuss Distance Metric Learning Using Proxies [70]	No	Yes	Yes	No	Yes	3
22	Deep Metric Learning via Facility Location [99]	No	Yes	Yes	No	Yes	3
25	PPFNet: Global Context Aware Local Features for Robust 3D Point Matching [17]	No	Yes	No	Yes	Yes	3
27	Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning [119]	Yes	No	No	Yes	Yes	3
1	Signature Verification Using A "Siamese" Time Delay Neural Network [7]	No	Yes	No	No	Yes	2
4	Distance metric learning for large margin nearest neighbor classification [127]	No	No	Yes	No	Yes	2
6	Quadruplet-Wise Image Similarity Learning [52]	No	No	Yes	Yes	No	2
7	Reidentification by Relative Distance Comparison [135]	No	Yes	No	No	Yes	2
8	Deep Metric Learning for Practical Person Re-Identification [132]	No	No	Yes	Yes	Yes	2
10	Improved Deep Metric Learning with Multi-class N-pair Loss Objective [98]	No	No	No	Yes	Yes	2
12	Deep Metric Learning via Lifted Structured Feature Embedding [100]	No	Yes	No	No	Yes	2

14	Learning Deep Embeddings with Histogram Loss [113]	Yes	No	No	No	Yes	2
24	Hard-Aware Deeply Cascaded Embedding [133]	Yes	No	No	No	Yes	2
26	Ranked List Loss for Deep Metric Learning [121]	No	Yes	No	Yes	No	2
29	Deep Metric Learning Meets Deep Clustering: An Novel Unsupervised Approach for Feature Embedding [73]	No	No	Yes	Yes	No	2
30	Exponential triplet loss [110]	Yes	Yes	No	No	No	2
2	Neighbourhood Components Analysis [27]	No	No	No	No	Yes	1
13	Deep clustering: Discriminative embeddings for segmentation and separation [34]	No	No	No	No	Yes	1
15	Local Similarity-Aware Deep Feature Embedding [37]	No	No	No	No	Yes	1

After reviewing over 30 publications in the field of DML, the following answers have been found to the research questions (RQ) defined in

- RQ1: In this study, 27 types of loss functions for DML have been identified. They have been categorized and listed in their historical order in 4. fig.. All the DML loss functions originate from Margin Ranking Loss, which itself is a variant of earlier Hinge Loss functions [127]. Then most of the newer loss functions originate from Contrastive loss [7], Triplet Loss [23], Histogram Loss [113], and Quadruplet Hinge Loss [52]. For most of the publications included in the study research subject is either the loss function itself or the sample mining methodology.

- RQ2: Latest loss functions and sample mining strategy achieve significantly better results than the previous functions as seen in 5. tab.. Also, datasets used in experiments have changed over time, but practical applications like image reidentification have not.

- RQ3: Most of the novel loss functions do not have theoretical explanations or derivations of the novel loss functions used in the model, but nonetheless some loss functions like Contrastive Loss [14], Triplet Hinge Loss [127], KISS-BCE Loss [46], Quadruplet Hinge Loss [52] and Binomial Deviance Loss [132] do have theoretical proof. Most of the other loss functions discussed in this study have their grounding in empirical experiments.

- RQ4: A number of significant limitations of DML loss functions and methods have been found in this study. Most of the loss functions require hyper-paramter α that is, a margin between clusters, but in realistic datasets this might not be equal for all classes. Some classes might have more variance than others. Some efforts have been made to resolve the issue like Proxy NCA Loss [70], but even this loss function requires hyper-parameter tuning and prior knowledge of class distributions.

  Intra-class similarities are also a significant problem. Most of DML loss functions ignore the fact that the same class samples also have their own distributions of similarities. Some works address this problem, but it still not fully solved [86] [52] [121].

  Sample mining strategies also are a major problem as they require significant computing resources dedicated just for selecting the best samples to train the model and apply the loss function. Multiple sampling strategies have been developed like Hard [23], Semi-Hard [23], N Hard Mining [98], Neighborhood Sampling [86], Distance weighted sampling [64] and others, but the problem is still not yet fully solved.

  Choice of the number of dimensions of embedding vectors and their embedding space also is a problem that still needs more studies. Publications differ in suggestions, how many dimensions to choose, and what normalization methods to apply to embeddings. Typical method is to use Euclidean distance with L2 normalized embeddings with high dimensionality of at least 128 dimensions [23], but some of the latest papers propose also alternative embedding space normalization [37] [117] and lower number of dimensions per embedding [110].

  Another significant limitation is the computing resources required to reach higher accuracy in reidentification tasks, some earlier works from 2015 required over 2000 CPU hours to reach the highest accuracy on face reidentification tasks [23]. Latest works have been using GPUs to accelerate and parallelize training, but even nowadays as datasets grow larger that requires expensive GPU hardware [110].

2.3 Conclusions of Literature Review

The results of the literature review indicate that the loss function design is an active research topic in DML. Many of the DML loss functions have been designed using empirical experiments instead of the derivation of classical mathematical theories.

DML loss functions come from Hinge loss and Ranking loss functions that have been around since the early days of computing. Then most of the modern loss functions originated from two major approaches, either by using triplets and Triplet Hinge Loss [127] or by using pairs and Contrastive Loss [7]. There has also been studies to use quadruplets or even more permutations of samples, but none has been as effective as Contrastive Loss and Triplet Loss based methods. The last group of DML methods are related to NCA [27] and other dimension reduction methods that produce similar outcomes like DML, but would not work as well for zero-shot learning settings.

Other significant areas of research have been identified for sample mining methods and embedding space normalization functions. Sample mining methods are necessary so that the model would use only difficult data samples in a loss function to improve the convergence speed of a loss function. The most common sampling methods are the Hard sample mining and the Semi-Hard sample mining [23]. Also, normalization of the embedding space is important, because the embeddings should be bounded to a predictable range of values to fill evenly the embedding space with clusters of different known and unknown classes. The most common method for normalization is to use Euclidean distance with L2 normalization, but some studies also have used cosine distances, Mahalanobis distance, angle in degrees, etc. DML is applicable to different types of practical solutions like reidentification of faces, speakers, signatures, handwriting, and product images.

3 IMPROVING THE PERFORMANCE OF FUNCTIONS USING DEEP LEARNING MODELS

This section of the thesis introduces a novel deep learning-based approach to optimize the performance of well-studied functions. As a practical application, the Value Iteration Algorithm has been used. It finds the shortest path from any position in the map to the target position in the map. Its performance degrades exponentially with the larger input size of the map as it cannot be executed in parallel, but novel iterative deep learning-based models can produce comparable results using parallelized architectures and achieve higher performance on larger maps.

The problem domain of Value Function and Value Iteration Algorithm learning has been described in 3.1. sec., then the existing Deep Learning based methods that can be used to model Value Function are described in 3.2. sec. and in 3.3. sec.. Finally, a novel method to model Value Function has been described in 3.4. sec.. The results of these methods have been shown in 6.1. sec..

3.1 Value Iteration Algorithm

Value Iteration Algorithm (VI) is used in classical reinforcement learning tasks to find an optimal policy for any problem within a fully observable environment. It can take into account the state transition model when the transition is uncertain [88]. VI is often used for finding the optimal path in maps with discrete states. A path finding task formalizes and discretizes a natural terrain and obstacles of the environment. Often this information is gathered using sensors that are attached to the mobile robot. These sensors might include LIDAR (Light Detection and Ranging), ultrasonic sensors for distance measurement, or IMU (Internal Measurement Units), etc. Discretization of a map is usually done, generating an occupancy grid. VI is an iterative algorithm that repeatedly applies the same Value function 1. eq. over all positions of a map to find the cumulative value of each cell position, as shown in 5. fig. Then the gradient between the values of these positions gives the policy of the optimal path. The policy of the optimal path enables an agent to find its way from any state in a discretized map to a positive terminal state. For each state s, there is a grid of positions and for each state

there are a number of actions a that can be taken, like moving up, down, left, right, or staying at a position. Depending on the environment, there could be more or different actions. The action a is chosen to maximize the cumulative reward with a given action R_a, then it is multiplied by transition model's probability P_a and added to adjacent state values $V(s')$ multiplied by discount factor γ like shown in 1. eq. Value function is called iteratively over the whole map until the values converge between iterations. The number of iterations needed to converge the value function grows exponentially with the size of the map. Convergence $|\frac{\overline{V(s)_i}}{V(s)_{i-1}}| < \delta$ in the context of VI is achieved when the relative difference between average values of states in a current iteration and previous iteration is smaller than δ. Depending on the cleanness of the data, the tolerance of the error and the type of task, δ could be between 0.1-0.001.

$$V_{i+1}(s) := \max_a \{\sum_{s'} P_a(s, s')(R_a(s, s') + \gamma V_i(s'))\} \tag{1}$$

Figure 5: Visualization of the Value Iteration Algorithm's consecutive iterations. Iterations depicted from the top left corner to the top right corner, then from the bottom left corner to the bottom right corner. After convergence has been achieved, it is possible to derive the optimal policy from every state to reach the terminal state with the highest cumulative reward. Green color represents the highest cell values (closest to the positive terminal state) and red color represents the lowest cell values.

The rationale behind modeling a value function using deep learning models is to increase its performance by parallelizing a task and reducing the sequential time complexity, as shown in 9. tab. There are other popular algorithms that use heuristics to reduce the time complexity, such as Dijkstra or A* [88], but they also reduce the precision of the result.

Table 9:

Comparison of proposed Value Itereation Solver Network (VSIN) with other methods.

Method	**Complexity**	**Quality**
VI (Value function)	$O(m \cdot n^2)$	Optimal
Dijkstra	$O(n^2)$	Good
A*	$O(h \cdot n)$	Medium
VISN (Proposed method)	$O(h \cdot n)$	Good

3.2 ConvNet and UNet models

To increase the performance of VI algorithm, the function can be modelled using a ConvNet model [56] [48] [97] [102] [31] [38] that has been trained to predict the output of $V(s)$ function. The whole model can be a Value function approximation. The ConvNet model works like filters for the whole occupancy grid map at once and predicts the output of VI without doing an iterative process. ConvNet also can be parallelized on modern GPUs and using deep learning frameworks, whereas VI is not a fully parallizable algorithm. The ConvNet models used in this research for the encoder part of VI are ResNet [31] and DenseNet [38] that are shown in 6. fig. The encoder of the model learns to compress and encode high-dimensional inputs to low-dimensional latent vector that later can be used in deeper parts of the model.

In this research, a decoder model with transposed convolution functions has been used [74]. The decoder model decompresses low-dimensional latent vectors to high-dimensional outputs that are applied as filters to the inputs. Architectures could be using simple arithmetical addition, multiplication, or substitution. Auto-Encoder model architecture that contains the decoder is shown in 7. fig.

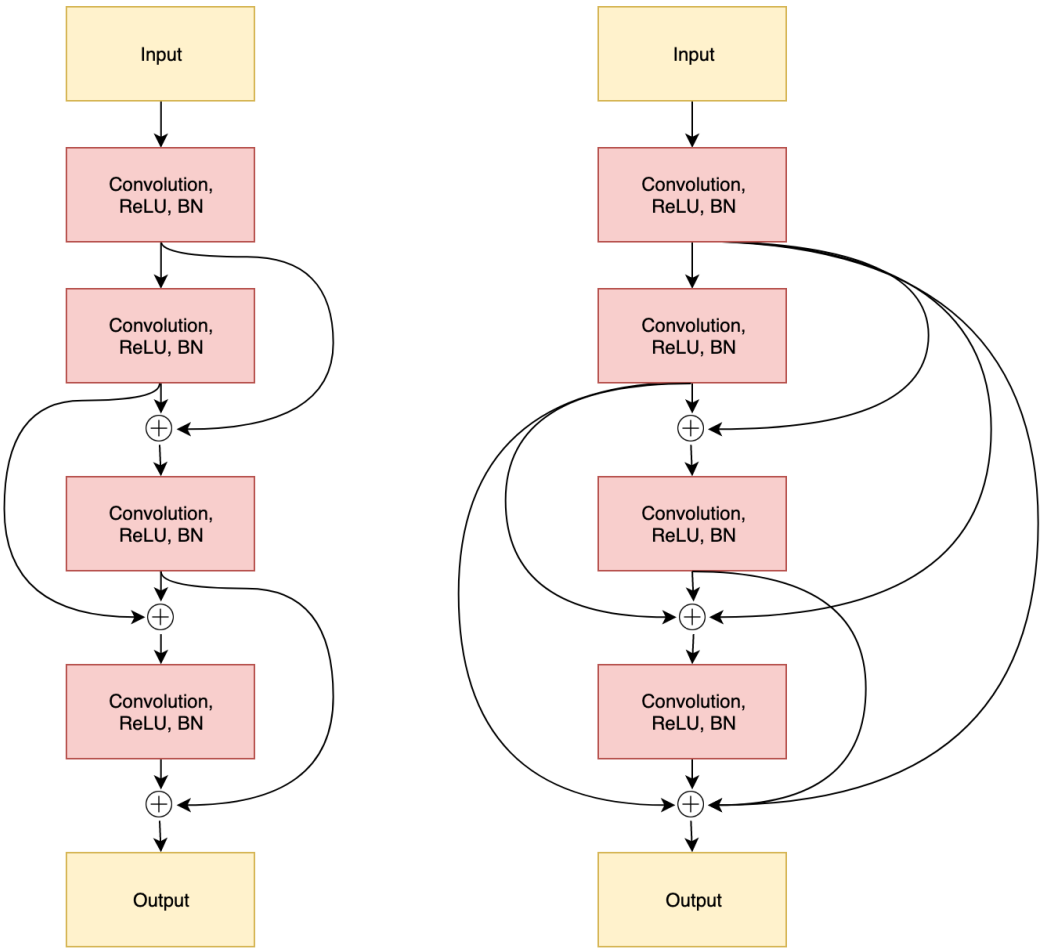

Figure 6: Comparison between ResNet on the left and DenseNet on the right encoder models. Plus sign denotes the arithmetic addition operation.

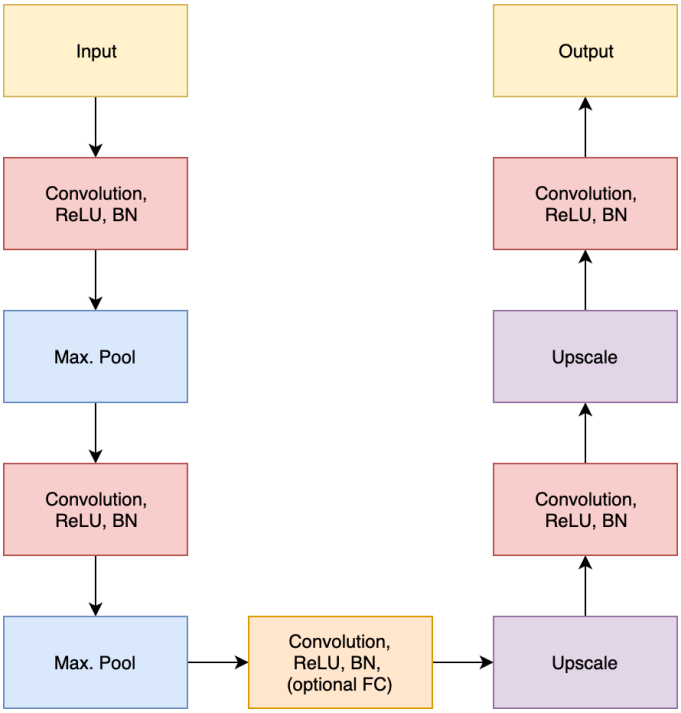

Figure 7: Example of Auto-Encoder model architecture. In the middle of the function graph, an optional affine transformation (FC) could be added if the models have fixed sized inputs and outputs.

The behavior of the VI algorithm is similar to the multi-pass filtering task as the structure of input data is not changed, but only fine details are modified by each pass. For filtering, style transfer tasks and even segmentation tasks, a good candidate is a UNet model [87] as it contains skip connections that maintain the details of the original input at different scales and depths of the model. Within this research, UNet models have been trained to obtain VI outputs in multiple iterations or also in a single step.

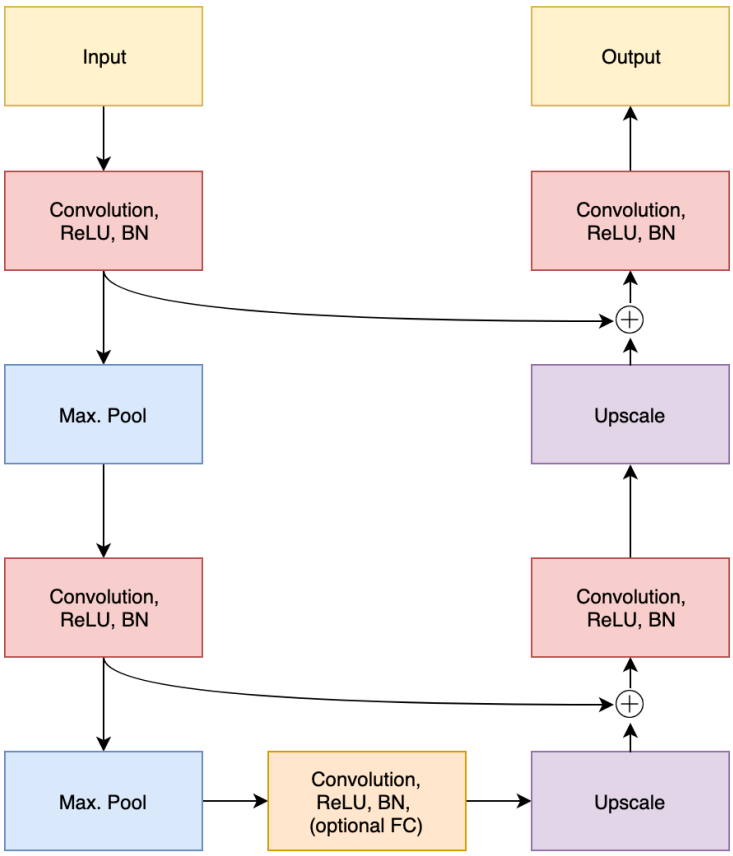

Figure 8: Example of UNet model architecture. Using skip connections, features are added from the encoder part of the function graph to the decoder part.

The original UNet model as depicted in 8. fig. contains the graph of functions as shown in equation 2. eq. In the equations, the input x is passed through a number of functions, where $Conv$ function is a linear 2D convolutional function with kernel size 3x3, stride 1 and padding 1 that will produce the same size output map. Whereas $DeConv$ are transposed 2D convolutional functions with kernel size 4x4, stride 2 and padding 1 that will produce twice as large output maps. Similarly, also $MaxPool$ functions are producing output maps twice as small as the input maps by choosing the maximum value of the reduced region. Skip connections are shown in 8. eq. and 11. eq. uses a concatenation operation, but for segmentation tasks it is often also used as an addition operation as in ResNet [102]. Final output y

that is limited by a sigmoid function σ that is then scaled to the range of values for every cell in the map.

$$o_1 = ReLU(Conv(x)) \tag{2}$$
$$o_2 = MaxPool(o_1) \tag{3}$$
$$o_3 = ReLU(Conv(o_2)) \tag{4}$$
$$o_4 = MaxPool(o_3) \tag{5}$$
$$o_5 = ReLU(Conv(o_4)) \tag{6}$$
$$o_6 = DeConv(o_5) \tag{7}$$
$$o_7 = (o_6, o_3) \tag{8}$$
$$o_8 = Dropout(ReLU(Conv(o_7))) \tag{9}$$
$$o_9 = DeConv(o_8) \tag{10}$$
$$o_10 = (o_9, o_1) \tag{11}$$
$$o_11 = ReLU(Conv(o_10)) \tag{12}$$
$$y = \sigma(o_11) \tag{13}$$

3.3 RNN models

Often, RNN (Recurrent Neural Network) models are used to model iterative processes and time-series type of data. In the case of VI algorithm, these models have been applied to reduce the complexity of a problem predicting all $V(s)$ at once by using a single iteration ConvNet model that has no knowledge of the history of previous timesteps. With RNN based model, the values are predicted in consecutive iterative steps, similarly how it is done by using VI algorithm, but unlike VI algorithm, the number of steps needed to converge the values is much smaller.

In this research, established RNN models like LSTM (Long-Short Term Memory) [36] and GRU (Gated Recurrent Unit) have been applied [28]. To improve the performance and speed of convergence, specific initialization strategies of parameters for these models were used. For example, initializing the bias vectors of the parameters of the forget gate function for LSTM model as scalar values of 1 to remember more information at the beginning of training. Similarly, initialize the biases of GRU model of the reset gate as a scalar value of -1 to achieve the same goal [41] [54] [25] [106].

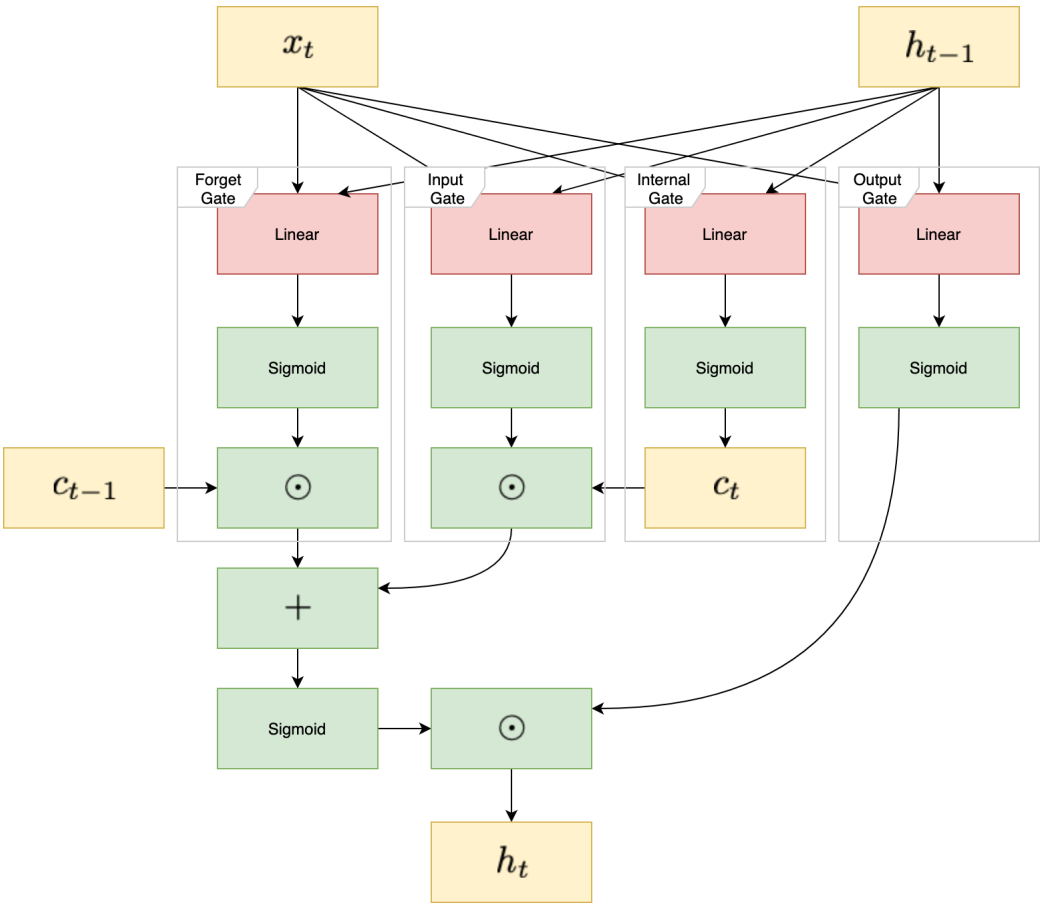

Figure 9: Function graph of LSTM model.

Function graph of LSTM model's equations are given below 14. eq. and visualized in 9. fig. Trainable weights and biases are denoted with W, U and b, sigmoid function σ based gates are f_t forget gate, i_t input gate, o_t output gate. Internal state c_t and hidden state h_t vectors in the beginning of each sequence are set to zero and they progressively with each sequence step are updated to a new value. At every time step, outputs of the cell are the current h_t value. According to the latest research, b_f should be initialized as $b_f = 1$ [106], RNN Dropout or ZoneOut regularization should be added [49] [94], h_0 and c_0 should be changed to learnable parameter just for the first timestep [75], Layer Normalization should be added for all linear transformations within LSTM cell [3] and, finally, multiple LSTM cells should be stacked upon each other should sum $h_t = \overset{layers}{\underset{k=i}{}} h_t^k$ similarly as it has been

done in ResNet for equal error propagation through all layers [102] [38] [106].

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{14}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{15}$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{16}$$
$$\tilde{c}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \tag{17}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{18}$$
$$h_t = o_t \odot \sigma_h(c_t) \tag{19}$$

GRU as shown below in 20. eq. and 10. fig. is a simplified version of LSTM with less learnable parameters. It contains only z_t update gate and r_t reset gate, and it uses not only sigmoid but also tanh ϕ functions as in traditional vanilla RNNs. To get the highest performance, the same optimizations as listed above for LSTM applies also to GRU, except b_r should be initialized to $b_r = -1$ [106].

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{20}$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{21}$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{22}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{23}$$

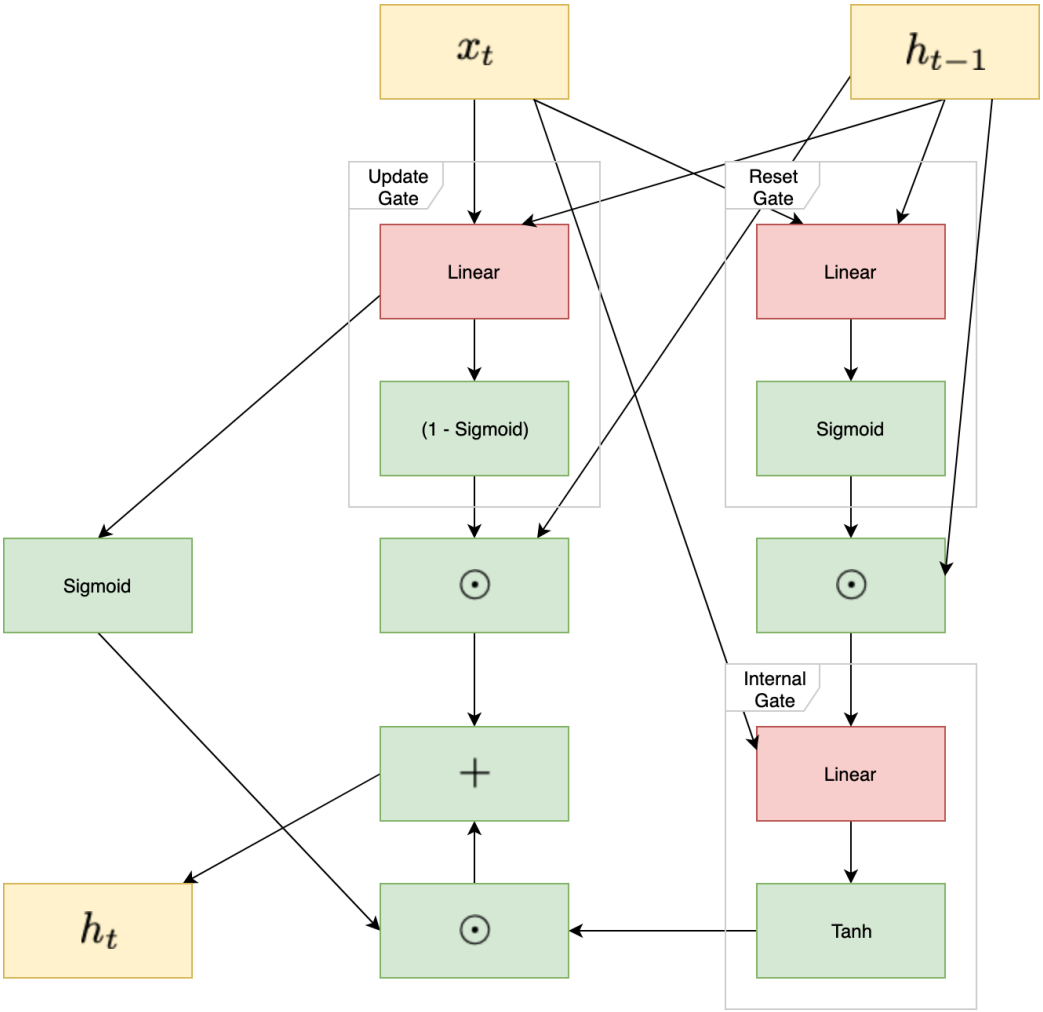

Figure 10: Function graph of GRU.

3.4 UNet-RNN-Skip model

As one part of the thesis, a novel UNet and RNN based model has been introduced. Novel UNet-RNN-Skip model contains multiple parts of the state of the art models put together with a purpose to improve the speed of convergence of the VI algorithm results. It contains UNet, LSTM, and similar skip connections to ResNet, but for time series and segmentation tasks.

UNet-RNN-Skip model is designed to approximate the value function

of VI algorithm in parallel on all cells in the occupancy grid, whereas VI algorithm needs to calculate the values only in sequential steps. It is also designed to approximate the value of a state in a single iteration, whereas VI algorithm would need multiple iterations to do the same task.

Another advantage of UNet-RNN-Skip model is that it can be trained on smaller maps with dimensions of 32x32 but then used in inference on much larger maps with dimensions of 256x256 without the need to retrain the model as it has learned the Value function itself and not only patterns of the map. All maps have been generated using a novel OccupancyMapGenerator algorithm that was also a part of this research and is described in 6.1. sec. Experimental results of properties of the model are listed in 6.1. sec.

The key of UNet-RNN-Skip model is to use LSTM at the bottom of "the U shape" of a function graph and at the same time use UNet skip connections similar to those used in ResNet, UNet++ [136] and UNet 3+ [39]. Unlike a simple UNet depicted in 8. fig. that uses concatenation, the new architecture uses arithmetic addition operations for skip connections. ResNet [31] function blocks depicted in 11. fig. have been configured in two different ways. First, there is an Identity ResBlock shown in the equation following 24. eq.

$$y_{c\times w\times h} = ReLU(Conv3x3(x_{c\times w\times h})) \tag{24}$$
$$y'_{c\times w\times h} = BatchNorm(y_{c\times w\times h}) \tag{25}$$
$$z_{c\times w\times h} = ReLU(Conv3x3(y'_{c\times w\times h})) \tag{26}$$
$$z'_{c\times w\times h} = z_{c\times w\times h} + x_{c\times w\times h} \tag{27}$$
$$u_{c\times w\times h} = ReLU(z'_{c\times w\times h}) \tag{28}$$
$$u'_{c\times w\times h} = BatchNorm(u_{c\times w\times h}) \tag{29}$$

Second, there is a BottleNeck which is shown in the equation 30. eq. The purpose of BottleNeck is to reduce the feature map size, but at the same time to increase the number of channels. ResBlock used here is similar to the preactivated ResBlock [32] by placing Batch Normalization before linear functions. The same functions have also been used for transposed convolutional layers where they are increasing the size of the output map and channel count.

$$y_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = ReLU(Conv3x3(x_{c\times w\times h})) \tag{30}$$

$$y'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = BatchNorm(y_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}}) \tag{31}$$

$$z_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = ReLU(Conv3x3(y'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}})) \tag{32}$$

$$x'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = Conv1x1(x_{c\times w\times h}) \tag{33}$$

$$z'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = z_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} + x'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} \tag{34}$$

$$u_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = ReLU(z'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}}) \tag{35}$$

$$u'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = BatchNorm(u_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}}) \tag{36}$$

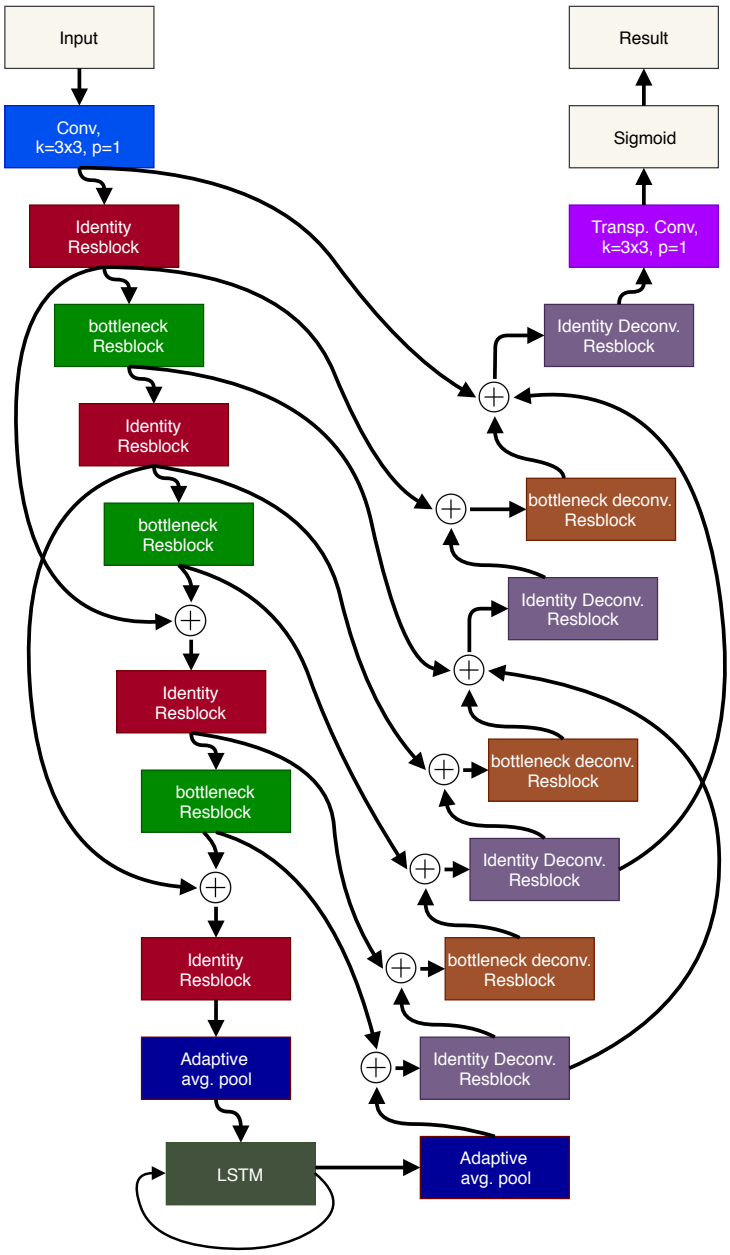

Figure 11: UNet-RNN-Skip model. Colors denote different blocks of functions used by the model.

4 FUNCTION SHAPING IN DEEP REINFORCEMENT LEARNING

This section of the thesis introduces a novel loss function and training procedure for Deep Q-Learning. The novel MDQN loss function uses the capabilities of dynamic function graphs by changing the shape of the loss function while the training is in the process. To study the properties of Deep Learning methods using Q-Value function, a survey of the methods has been done. Novel procedures also provide visualizations of policies and values of the states to change black box models into white box models.

The problem domain of Deep Learning-based reinforcement learning has been described in 5.1. sec., then the existing DQN loss function has been described in 4.2. sec. and finally, a novel loss function has been described in 4.3. sec.. The results of these methods have been shown in 6.2. sec..

4.1 Q-Value and policy gradient functions for reinforcement learning

Reinforcement learning algorithms have been well developed even before the advent of Deep Learning, but with new methods their capacity has greatly increased [68]. Similarly, like the previously discussed VI algorithm, these algorithms are used to estimate the best policy π at a given state s_t.

Three major sets of methods exist in classical reinforcement learning:

1. Q-Value based methods
2. Policy gradient methods
3. Actor-Critic methods (combination of Q-Value and Policy gradient methods).

All of these sets of methods mostly are grounded in loss function shaping, as it is one of the most important parts of the algorithm to make it work.

Policy gradient methods rely on the policy gradient theorem, 37. eq. where the gradient of error for the policy function π is the estimation of probability of trajectory of previous states τ at state s_t and action a_t multiplied by cumulative reward R 38. eq. of this trajectory τ. This gradient function is unstable and too hard to converge, but more recent advances

that came with Deep Learning like TRPO [92], PPO [91], MERLIN [125] and IMPALA [21] have improved it by a large margin.

$$\nabla \pi_\theta(a_t|s_t) = \nabla log \pi_\theta(a_t|s_t) R(\tau) \tag{37}$$

Cumulative reward R 38. eq. is the total reward of the trajectory of states, multiplied at each time step t by a discount factor γ. A small discount value γ prioritizes short-term rewards, but a larger value prioritizes rewards and positive events that happen at later stages of an episode. Discount factor usually is a hyper-parameter set at a value in the range 0.9-0.99, but for environments where the reward comes later in an episode it might have a lower value.

$$R = \sum_{t=0}^{n} \gamma^t r_t \tag{38}$$

The second set of methods is grounded in the Bellman equation 39. eq. that estimates R the value using a model of the Q-Value function for a given state s_t when action a_t has been taken. By examining and selecting an action a_t that is giving the highest Q-Value, it is possible to determine the policy π.

$$Q_\pi(s_t, a_t) = r_t + \max_{a'} Q_\pi(s_{t+1}, a') \tag{39}$$

Third set of methods are Actor-Critic methods that are grounded in a combination of two previous methods. Many successful variants of these methods have been developed like DDPG (Deep Deterministic Policy Gradient) [60], A3C (Asynchronous Advantage Actor-Critic) [67], GPU A3C [4] and ACER (Actor-Critic with Experience Replay) [124].

Although it might seem reasonable that the best results should be achieved using a later set of methods, the state of the art results have been achieved mostly with the first two methods used alone.

4.2 Deep Q-Network model and Loss function

One of the most successful models for Deep Reinforcement learning has been DQN (Deep Q-Network). Since the initial success of DQN [68], there has been a significant development of Q-Value function-based approaches [2]. Initial DQN was tested on Atari games that have only very high-dimensional state

representations - either RAM memory dump or raw pixels of each frame of a game state as shown in 12. fig. Another common environment for evaluating RL (Reinforcement Learning) models is the OpenAI Gym [6]. However, in contrast to Atari games, these environments often contain only simple tasks and have only low-dimensional state representations. For example, MoonLander environment in the state representation includes the position, speed, and angle relative to the target location on the moon's surface. In this research, PLE (PyGame Learning Environment) has been used for validating models [104]. It contains high and low dimensional state representations for each environment, and it is even possible to modify the behavior of the environment while the model is trained to better study the properties of a model.

Some small, but significant additions over the last years have been made to DQN to substantially improve its performance. One of the most important improvements is the Prioritized Experience Replay [90] that enables sampling the most valuable samples with the highest TD (Temporal Difference) loss to be selected more often for training. Another improvement is Dueling DQN [123] that tries to enforce the model to learn instant and delayed rewards, primarily using the specific architecture of the model. The next improvement is DDQN (Double DQN) [30] [29] that allows for more stable convergence of DQN based loss function 40. eq. by introducing new DDQN loss function 41. eq. In DDQN model, the weights Q_{target} are copied and frozen from Q_Θ every predefined time interval during process.

$$\mathcal{L}_{dqn} = \begin{cases} (r_t + \gamma \max_{a'} Q_\Theta(s_{t+1}, a') - Q_\Theta(s_t, a_t))^2 & if\ t < t_{last} \\ (r_t - Q_\Theta(s_t, a_t))^2 & if\ t = t_{last} \end{cases} \tag{40}$$

$$\mathcal{L}_{ddqn} = \begin{cases} (r_t + \gamma \max_{a'} Q_{target}(s_{t+1}, a') - Q_\Theta(s_t, a_t))^2 & if\ t < t_{last} \\ (r_t - Q_\Theta(s_t, a_t))^2 & if\ t = t_{last} \end{cases} \tag{41}$$

Figure 12: Atari's games were used for DQN based model evaluation [68].

Finally, Rainbow-DQN has shown that when all small additions are put together, the model outperforms all of these additions alone [35] as shown in 13. fig. In the chart it shows a comparison of different loss functions and methods regarding the millions of frames it took to train them and normalized the score across all Ataris games. Rainbow-DQN is based on DDQN loss function with the additions mentioned above.

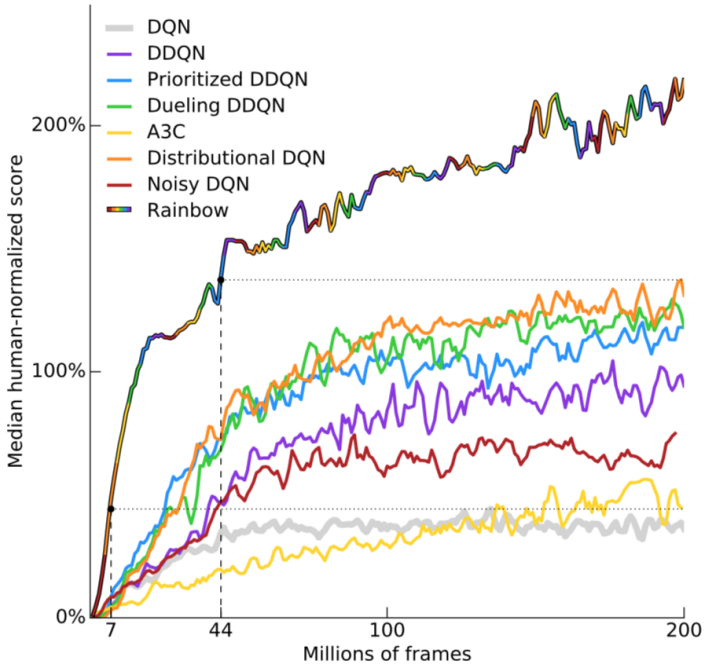

Figure 13: Rainbow-DQN comparison to its parts using normalized Atari's score [35].

4.3 Multi Deep Q-Network model and Loss function

The thesis introduces a novel MDQN (Multi Deep Q-Network) loss function, which is a dynamic loss function that changes its behavior while the model is training. Similarly to DDQN, it contains target models that are updated with predefined time frame intervals, but unlike DDQN, there can be more than one additional version of DQN model. Two or three additional versions of DQN model can be changed intermittently to dampen the effects of short-term events on the Q-function and stabilize the learning process. The behavior of MDQN loss function is listed in the pseudocode of Algorithm 1. alg. Depending on MDQN variant, multiple weights of DQN models $Q_1, Q_2, ..., Q_n$ are initialized in the beginning. For example, MDQN-3 contains 3 parallel Q models. Then initial Q_a and Q_b are randomly sampled from a set of DQN models. Frame counters $c_a, c_b, ..., c_n$ are set to zero. Hyper-parameters

$threshold_a, threshold_b, ..., threshold_n$ are found using grid search, but they also could be learnable parameters that would be used only for a training procedure. MDQN also uses state transition sets a_t, s_t, s_{t+1}, R_t that are similar to those seen in a replay buffer to train Q-Value functions using the actual ground truth values of the cumulative reward. Not only transition tuples are easily matched in replay memory when using low-dimensional environments in PLE (PyGame Learning Environment) [104], but also can be found in high-dimensional (pixel space) environments via Deep Metric Learning and embedding vector similarities that are described in the next chapter of this thesis.

Algorithm 1: MDQN loss function

procedure TRAIN
  $Q_1, Q_2, ..., Q_n$
  $threshold_a, threshold_b, ..., threshold_n$
  $c_a, c_b, ..., c_n = 0$
  $Q_a = Sample(Q_1, Q_2, ..., Q_n)$
  $Q_b = Sample(Q_1, Q_2, ..., Q_n)$
  while $Training = True$ **do**
    for do$\{a_t, s_t, s_{t+1}\}$ sample from $ReplayBuffer$
      $\forall n, c_n = c_n + 1$
      if $\forall n, c_n > threshold_n$ **then**
        $Q_b = Q_a$
        $Q_a = Q_n$
        $c_n = 0$
      if $\{a_t, s_t, s_{t+1}\}$ similar exist in $ReplayBuffer$ **then**
        $Q_a(a_t, s_t) \leftarrow \sum_{t=0}^{t+1} \gamma^t R_t$
      else
        if $s_t \neq$ terminal state **then**
          $Q_a(a_t, s_t) \leftarrow R_t + \gamma \max_a Q_b(a, s_{t+1})$
        else
          $Q_a(a_t, s_t) \leftarrow R_t$
    while $s_t \neq$ terminal state **do**
      $a_t \leftarrow max_a average(\{Q_a(a, s_t), Q_b(a, s_t)\})$
      ...
      store $\{a_t, s_t, s_{t+1}, r_t\}$ in $ReplayBuffer$

5 FUNCTION SHAPING IN DEEP METRIC LEARNING

This section of the thesis introduces a novel Exponential Triplet loss function and novel training procedures for deep metric learning. It also introduces novel embedding space normalization functions that provide cosine distance properties to Euclidean distances.
The problem domain of reidentification task is described in 5.1. sec., then the existing Deep Learning loss function to solve the reidentification task is described in 5.2. sec. and finally, a novel loss function is described in 5.3. sec.. The results of these methods have been shown in 6.3. sec..

5.1 Zero-Shot learning and Re-identification task

Zero-Shot learning is a subset of machine learning methods that are based on a model that can be applied for the categorization of novel classes in the inference phase as shown in 14. fig. These novel classes have never been seen in a model during the training phase [66] [23] [44] [62]. Usually, both datasets in the training and inference phase are from the same domain, for example, dataset of photos of faces, voice recording dataset, photos of cars dataset, etc.

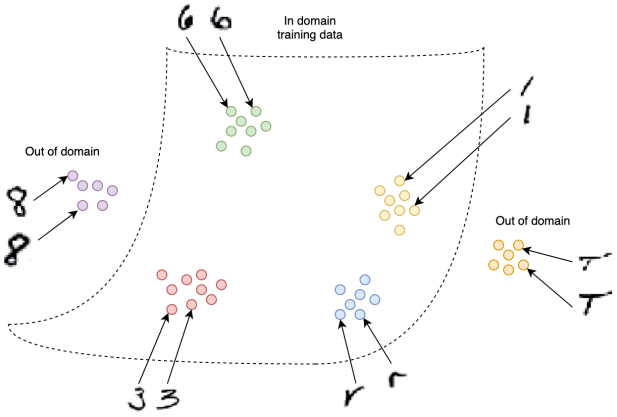

Figure 14: Zero-shot learning visualization using EMNIST dataset with a split of disjoint training and testing datasets.

The advantage of such models is that it is not necessary to retrain them

for new classes, and the characteristics of the model can be fine-tuned after the training phase. This can be achieved by changing the thresholds and parameters of clustering algorithms, as well as a latent vector space normalization in the inference phase. Another advantage is that for novel classes, very few data points are needed instead of thousands of data points normally needed in classification models that do not use zero-shot learning methods [116]. Zero-shot learning has high sample efficiency that can have comparable accuracy to full training of a dataset with even just 2 samples [116]. Sometimes in the scientific literature, zero-shot learning is also called one-shot learning, although technically one-shot learning means that during the training process the model is allowed to see at most one sample from each class. There also exists few-shot learning where multiple data samples of new classes are given during the training process of the model. These classes can be given alone or along with other classes that contain many more samples. Usually these methods are used with transfer learning to adapt the model for novel classes, but in zero-shot learning the model has never seen any of the target classes during the training. After training, the model is an encoder that is able to compress high-dimensional information like photos down to low-dimensional latent vectors that maintain semantic information. Distance between such vectors ensures clustering of the novel and existing classes, thus these models are often labeled as Deep Metric Learning (DML) or Distance Metric Learning. These models are widely used for the reidentification task for face verification and other biometric verification systems [23] as shown in 15. fig. In such systems, each person is enrolled with one or multiple samples of their face's photo that becomes a novel class of dataset. Later, the system can reidentify this person using any other photo never seen in the enrollment or the training process.

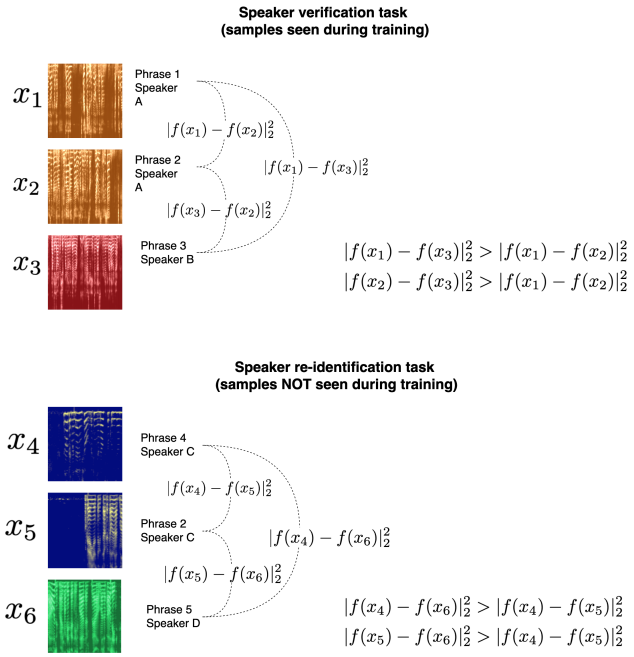

Figure 15: Difference between the speaker verification task and speaker reidentification task (implemented in "asya" commercial system). Re-identification task is a zero-shot learning, because model f has not seen audio samples x_4, x_5, x_6 during the training.

5.2 Triplet Loss Function

For the model to learn deep representations or embedding of input data in a zero-shot case, usually either Contrastive Loss [8] or Triplet Loss [23] based functions are used. The goal of the Triplet Loss function 42. eq. is to increase the distance of sample vectors from different classes $\|y_a - y_n\|_2^2$ and to decrease the distance of the same class sample vectors $\|y_a - y_p\|_2^2$, as shown in 16. fig. At the same time, the function is built to not collapse the same class vectors into a single modality, but to have the margin distance α between them. Often for distance metrics, cosine or Euclidean distance is used.

$$\mathcal{L}_{std} = |\|y_a - y_p\|_2^2 - \|y_a - y_n\|_2^2 + \alpha|_+ \tag{42}$$

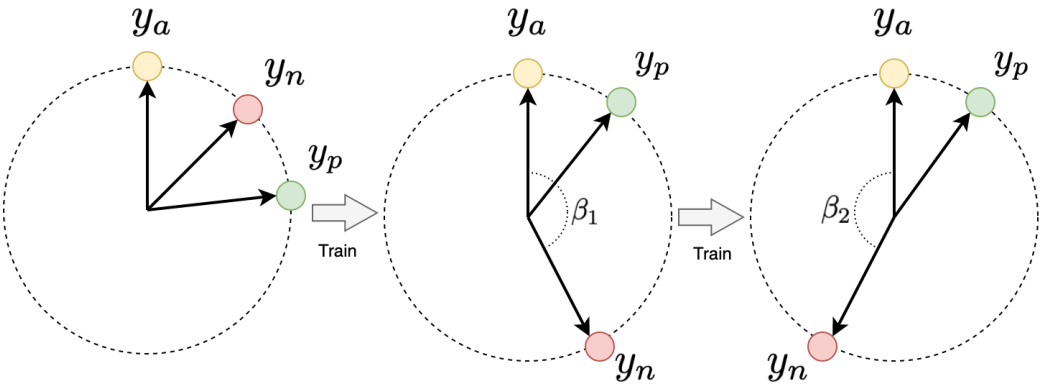

Figure 16: Triplet loss-based training 42. eq. using cosine distances of embedding anchor y_a vector, same class as anchor y_p vector, different class y_n vector. When a negative pair is pushed to a maximum distance, y_n will start to become closer again to y_a.

Cosine distance is preferred because it has a cyclic nature. With the cosine distance, when the sample distance is larger than the maximum distance of 2, it will return to 0. Triplet loss-based functions normally require to have very selective sample mining algorithms and filtering constraints on what samples can be allowed to pass to the loss function. For example, usually during training, the worst pairs of samples are found for the loss functions. Worst positive, the same class pairs are those with the longest distance in between them and the worst negative, different class pairs are those with the shortest distance in between them. These kinds of pairs regarding the anchor point y_a gives the largest error gradient and helps to converge the loss function faster.
Model $f_\theta(x)$ is an encoder that reduces dimensionality from high dimensional input image x to low dimensional embedding y. Embedding should be chosen with 32 dimensions or larger, as explained in 8. sec. [110]. Parameters θ are the same for every sample x, as shown in 17. fig.

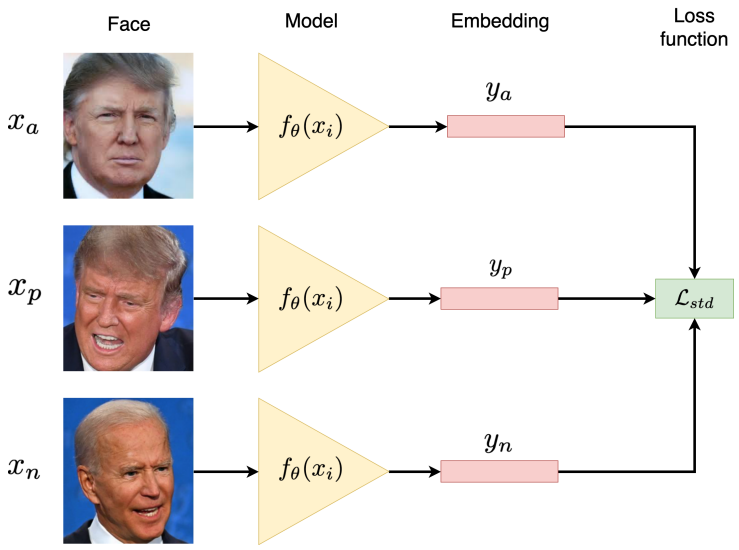

Figure 17: Example of a triplet loss applied for a face reidentification task. Encoder $f_\theta(x_i)$ share the weights. Dimensionality of y_i is much smaller than x_i.

5.3 Exponential Triplet Loss Function

In the thesis, a novel Exponential Triplet Loss function $\mathcal{L}_{exp}$ has been introduced that has a specifically designed shape of error space as seen in 18. fig., which leads to better convergence than a Triplet Loss $\mathcal{L}_{std}$ function described in the previous section.
Exponential Triplet Loss $\mathcal{L}_{exp}$ has asymmetric shape that keep negative pairs not closer within a half of maximum distance $\max(f_{emb}(x))$ of embedding space, normally, $\max(f_{emb}(x)) = 2.0$ when it is in a spherical space.
c_n is the minimal class cluster size similar to margin α in $\mathcal{L}_{std}$. This distance c_n is calculated by dividing the maximum distance $\max(f_{emb}(x))$ by the number of classes K in the training dataset 43. eq.

$$c_n = \frac{\max(f_{emb}(x))}{K} \tag{43}$$

$$emb_p = \frac{\|y_a - y_p\|_2^2}{\max(f_{emb}(x))} \qquad emb_n = \frac{\|y_a - y_n\|_2^2}{\max(f_{emb}(x))} \tag{44}$$

$$\mathcal{L}_{exp} = -log(1.0 - \frac{|emb_p - c_n|_+}{1 - c_n} + \epsilon) - log(1.0 - \frac{|0.5 - emb_n|_+}{0.5} + \epsilon) \quad (45)$$

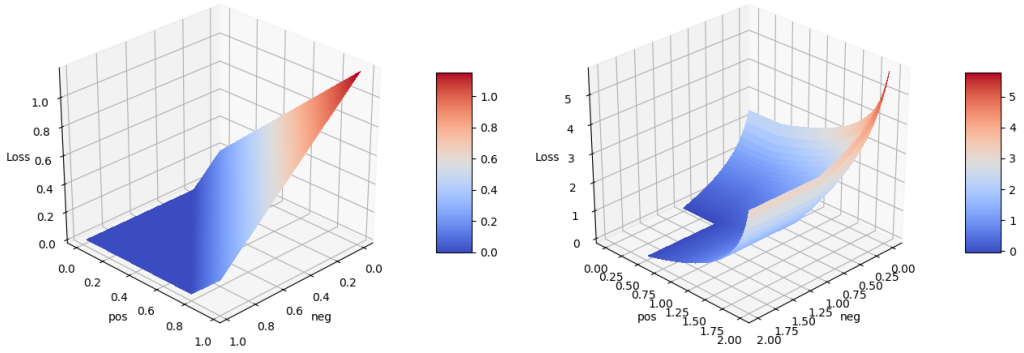

Figure 18: Comparison between $\mathcal{L}_{std}$ and $\mathcal{L}_{exp}$ functions. Positive pair distance $\|y_a - y_p\|_2^2$ (pos) and negative pair distance $\|y_a - y_n\|_2^2$ (neg).

To further improve the performance of L_{exp}, multiple other loss functions from recent research has been combined into the composite loss function $Lcomp$, 49. eq. Additions include L2-constrained Softmax with cross-entropy $\mathcal{L}_{class}$ [84] and center loss $\mathcal{L}_{center}$ [128] 47. eq. 48. eq. Within $\mathcal{L}_{class}$ 46. eq. input in Softmax function $f(x)$ is L2 normalized and scaled by s. Within $\mathcal{L}_{center}$ during training, class instances are accumulated and then c_{y_i} the center of the cluster is calculated.

$$\mathcal{L}_{class} = -\sum_{i=1}^{M} y_i log \frac{e^{W_i^T s|f(x_i)|_2^2 + b_i}}{\sum_{j=1}^{C} e^{W_j^T s|f(x_i)|_2^2 + b_j}} \quad (46)$$

$$\mathcal{L}_{center'} = \sum_{i=1}^{M} ||x_i - c_{y_i}||_2^2 \quad (47)$$

$$\mathcal{L}_{center} = \sum_{i=1}^{M} |||x_i - c_{y_i}||_2^2 - \frac{c_n}{2}|_+ \quad (48)$$

$$\mathcal{L}_{comp} = \mathcal{L}_{exp} + C_{center}\mathcal{L}_{center} + C_{class}\mathcal{L}_{class} \quad (49)$$

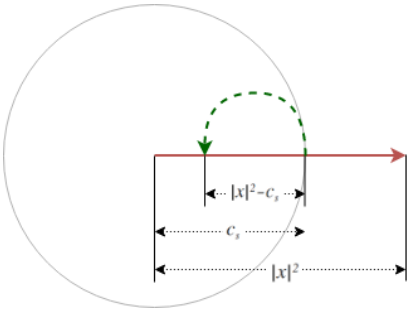

Figure 19: Illustration of Unit-Bounce embedding normalization function within L2 spherical space.

Another contribution to DML is a function to normalize an embedding space called Unit-Bounce 50. eq., 51. eq. It has similar properties to the cosine distance in the Euclidean space. As shown in 19. fig. when the embedding vector reaches the edge of a sphere, it bounces back towards the center of the embedding space, and when the embedding reaches the opposite side of the sphere with a radius of c_s it bounces back towards the center again. This ensures that the whole 3D latent space is effectively used instead of just the surface of the sphere as it is in the case of L2 normalization.

$$f'_{emb}(x) = \begin{cases} f_{bounce}(x), & \text{if } |x|^2 \geq 1 \\ x, & \text{otherwise} \end{cases} \tag{50}$$

$$f_{bounce}(x) = \begin{cases} |x|^2 - \left\lfloor \frac{|x|^2}{c_s} \right\rfloor - c_s \frac{x}{|x|^2}, & \text{if } \lfloor \frac{|x|}{c_s} \rfloor \bmod 2 = 0 \\ c_s \frac{x}{|x|^2} - |x|^2 - \lfloor \frac{|x|^2}{c_s} \rfloor, & \text{otherwise} \end{cases} \tag{51}$$

6 EXPERIMENTAL RESULTS AND APPLICATIONS

This section of the thesis lists the main experimental results of the novel loss functions and methods described in 3. sec., 4. sec. and 5. sec.. Novel methods have been tested in multiple domains starting from the task of VI algorithm, Deep Reinforcement learning in various computer game environments, Deep Metric learning for face reidentification tasks, and ending with practical applications in analytical chemistry and humans voice processing. The full results of the studies described in this section are listed in scientific publications Appendix A, Appendix B, Appendix C, Appendix D, Appendix E

6.1 Results of UNet-RNN-Skip model

UNet-RNN-Skip model has been tested in the problem set of VI algorithm to optimize the speed of convergence of the Value function. VIN (Value Iteration Network) is a model based on UNet-RNN-Skip that is usable for different map sizes as shown in 20. fig. without need to retrain the VIN for each of the map sizes. VIN iteration count to achieve convergence grows linearly with a map size, whereas VI algorithm's execution speed to achieve convergence grows exponentially when increasing a map size. The metric success rate has been used to determine stability of the model on the test dataset. The success rate is a percentage that describes the number of cells on the map that contain a path to the positive terminal state via the gradient of the values of adjacent cells. For VI, the metric of success rate will always be 1.0 after the convergence. The research also established that for the task of VI algorithm, UNet models outperform convolutional AE models, as shown in 10. tab.

Table 10:

Comparison of ConvNet based and UNet based models for VI problem.

Model	Loss	Success rate	Epoch (min)
Conv-AE-RNN	8.58E-06	0.598	10.862
UNet-RNN-Skip	3.04E-06	0.998	15.833

Table 11:

Comparison of VIN and VI methods for speed (sec.) to convergence.

Model / Map Size	32	64	128	256
VI	2.95	24.873	195.902	1473.108
VIN	0.031	0.071	0.236	0.833

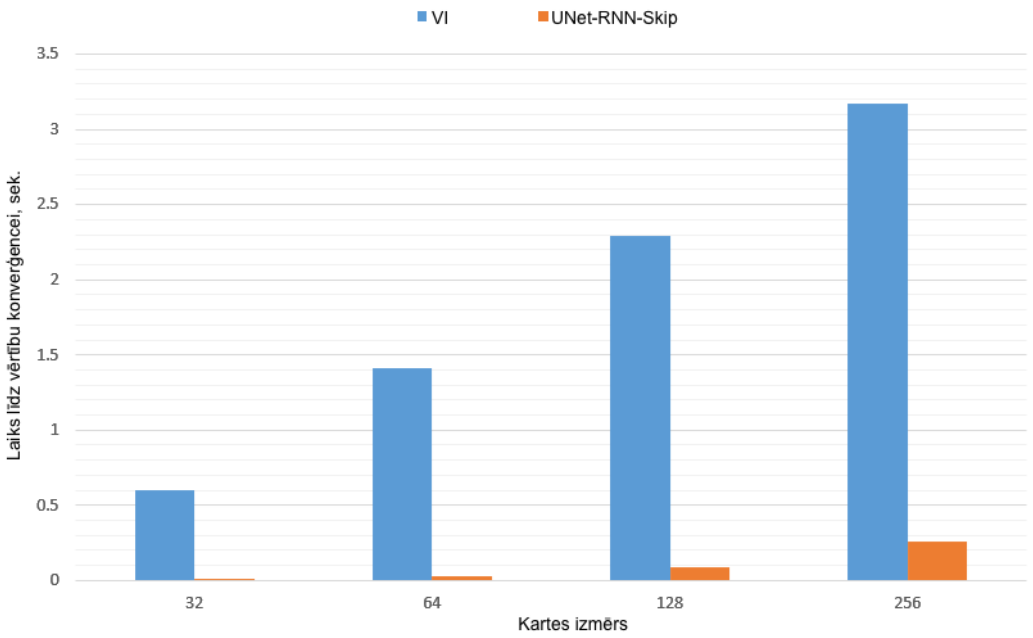

Figure 20: Comparison of convergence speed VI versus VIN based on UNet-RNN-Skip.

With the novel UNet-RNN-Skip model, this research work introduced also the synthetic dataset generator OccupancyMapGenerator for occupancy grid with obstacles and mazes as shown in 21. fig..

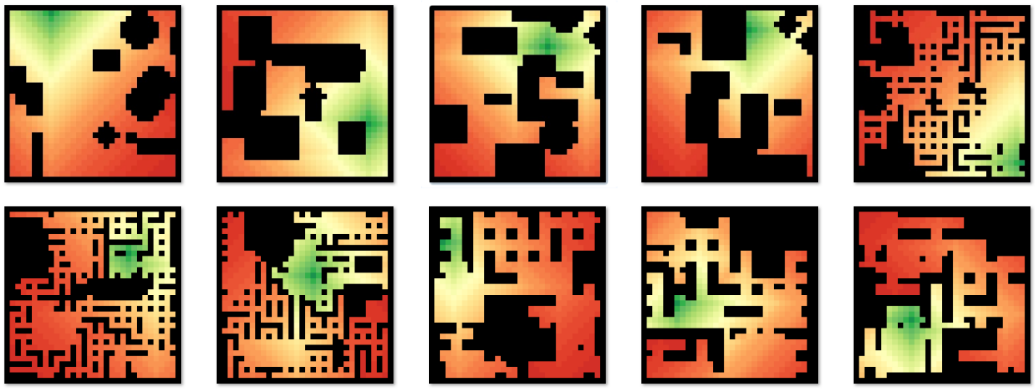

Figure 21: Examples of synthetic maps generated by OccupancyMapGenerator and cell values to reach the positive terminal state (Green - the highest value, Red - the lowest value).

The generator is capable of generating any predefined size 2D occupancy grid map using command line arguments. It produces maps in PNG image format and executes VI algorithm on the maps and stores the optimal state values. The generator uses predefined constants of coverage percentages of obstacles. Obstacles consist of a randomly generated maze, rectangles, and circles. Dijkstra path-finding algorithm [18] is used to check the reachability of each cell to ensure that all walkable cells are interconnected with each other. For maze generation, a Recursive Backtracking algorithm has been used. The algorithm of OccupancyMapGenerator is listed below in 2. alg.

Algorithm 2: OccupancyMapGenerator map generation algorithm

procedure GENERATEMAP
    $size$
    $types\_obstacles = \{maze, circles, rectangles\}$
    $max_{coverage}$
    $iterations\_obstacles$
    ϵ_{VI}
    $M_{size \times size} \leftarrow generateZeros(size)$
    if $maze \in types\_obstacles$ **then**
        $M \leftarrow generateMaze(M)$
    if $circles \in types\_obstacles$ or $rectangles \in types\_obstacles$ **then**
        for $iterations\_obstacles$ **do**
            $coverage = \frac{walkable}{size^2}$
            if $coverage < max_{coverage}$ **then**
                $M \leftarrow generateObstacles(M, types\_obstacles)$
            else
                break
    $goal_{x,y} \leftarrow RandomWalkablePosition(M)$
    for $pos_{x,y}$ in M **do**
        if $pos_{x,y} \in walkable$ **then**
            $reachable \leftarrow dijkstra(M, goal_{x,y}, pos_{x,y})$
            if $\neg reachable$ **then**
                $M \leftarrow fillHole(M, pos_{x,y})$
    $M_{vi} \leftarrow valueIteration(M, \epsilon_{VI})$
    $store(M_{vi}, M)$

6.2 Results of Multi Deep Q-Network Loss Function

MDQN loss function [109] has been tested in multiple computer game environments in PLE (PyGame Learning Environment) [104].
It has been tested on games like Flappy Bird, Pong, 3D Raycast Maze, and VizDoom as shown in 22. fig. These games provide high-dimensional states such as the raw pixel matrix, and some of them, like Pong, also provide low-dimensional state that is especially useful for quickly validating novel loss functions before applying them to a high-dimensional input.
As seen in 12. tab., MDQN loss function achieved higher performance that DDQN loss function that at the time of publication was state of the art approach for Deep Q-Learning based reinforcement learning [30].
Grid search of hyper-parameters and 20 repetitions of training procedures were done for all combinations of environments and loss functions to ensure fair comparison between methods.

Figure 22: Example of PLE games. Top row left "Pixel Helicopter", then "Monster Kong". On the right there is "Flappy Bird". In the bottom row, 2 variants of VizDoom mini-games that have also been trained within this research (Video of an agent that is trained using MDQN loss function: `https://www.youtube.com/watch?v=oqN6rtnv1EI`)

MDQN loss function outperformed DDQN loss function in PLE environments and was more robust when experiments were repeated multiple times, as shown in 12. tab.
Along with MDQN loss function, a new method for visualizing Q-function values of each state has been developed. A new Q-Value Map is obtained by manipulating objects within a computer game environment to produce an accurate understanding of an agent's policy at every time step and every stage of learning. As these games have open-source code in PLE, for example, it is possible to manipulate the location of the player and then calculate Q-value for every pixel or set of states in the frame as shown in 23. fig.

Table 12:

Results of the average score for 20 experiments using different loss functions in PLE.

Loss Function	Environment	Avg. Score
MDQN	Flappy Bird	17.2
DDQN	Flappy Bird	16.9
MDQN	Pong	1.7
DDQN	Pong	1.3
MDQN	3D Raycast Maze	3.9
DDQN	3D Raycast Maze	3.7

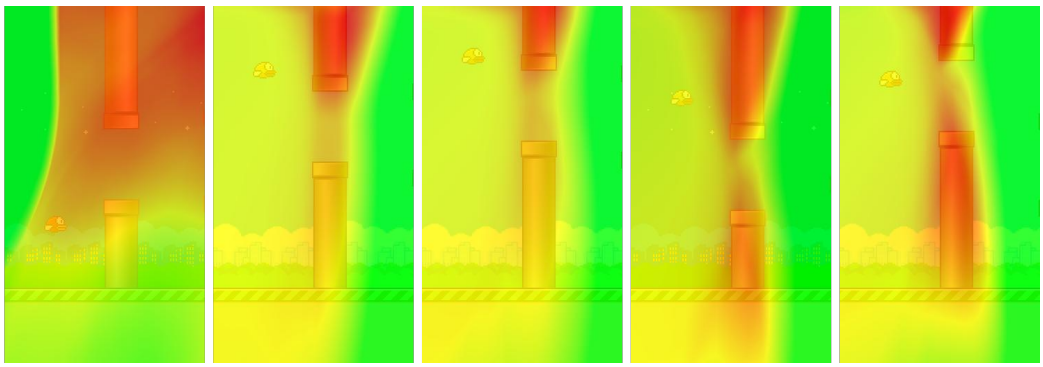

Figure 23: Q-Value Map of FlappyBird environment, where each pixel represents a Q-Value if the player (bird) would be located in this position. Red color denotes low Q-Value and Green color denotes high Q-Value. On the left there are Q-Value maps before training, in the middle during the training, and on the right after the training.

Even for 3D environments such as 3D raycast maze, it is possible to evaluate each state in the map by rotating the camera around the Z axis and averaging Q-Value from rendered pixels in 360 degrees. Then the average values can be plotted on top-down view as shown in 24. fig. As shown in, 13. tab. MDQN function achieved the highest average score over 20 training repetitions and the least variance of the score compared to DDQN and DQN functions. MDQN-2 and MDQN-3 denote 2 and 3 sets of copies of MDQN models used in MDQN function.

Table 13:

Results for 20 training repetitions using different loss functions in 3D raycast maze environment.

Loss Func.	Learning Rate	Avg. Score	Var. Score
MDQN-2	1.00E-05	3.904359232	0.728045918
DQN	1.00E-05	3.88654262	2.124993494
MDQN-2	1.00E-06	3.7166532	0.154117942
DDQN	1.00E-06	3.713829593	1.524318234
DDQN	1.00E-05	3.638360789	1.662039807
DDQN	0.0001	3.267777864	2.889255991
MDQN-3	1.00E-05	3.056116361	2.339890159
DQN	1.00E-06	3.026868771	2.028895348
MDQN-3	1.00E-06	2.770128326	0.714132328
MDQN-2	0.0001	2.545370799	4.120312752
DQN	0.0001	2.24425396	2.153779645
MDQN-3	0.0001	2.174641347	3.541216037

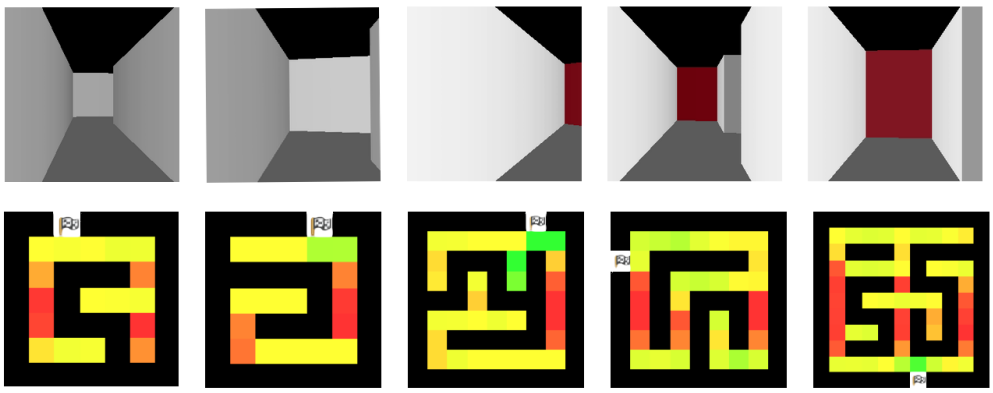

Figure 24: In a top row of frames of the 3D raycast maze environment. In a bottom row Q-Value Map of the environment from top-down view of the 3D raycast maze before, during, and after training.

Finally, along with the novel MDQN loss function, an extensive survey of Deep Q-Learning based methods has been done. In the result, it has been found that Deep Q-Learning based methods are very sensitive to the random seed and intrinsic randomness of the environment on which they are tested. As shown in, 25. fig. a sufficient number of repetitions are needed to find a sample with the highest score using the same set of hyper-parameters. It has been found that 20 repeated full training procedures are necessary to establish an accurate baseline of the performance of these methods.

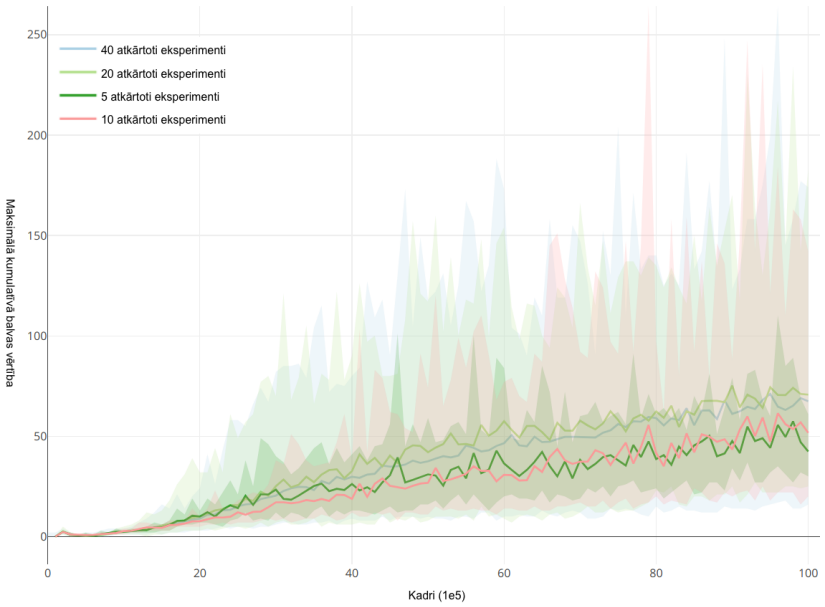

Figure 25: Variance of the score of the same hyper-parameters with different number of re-training procedures of Deep Q-Network in Flappy Bird environment.

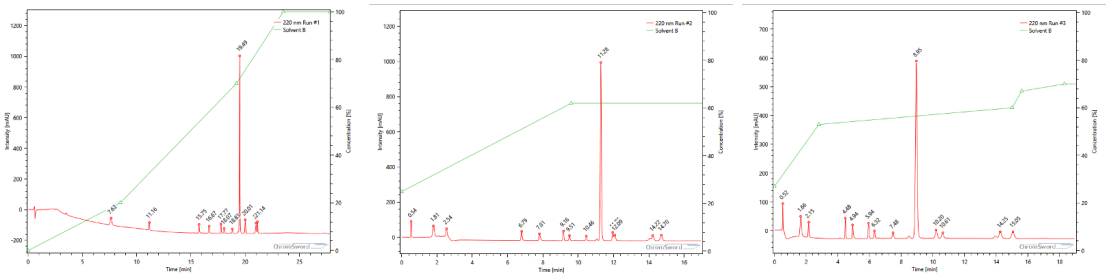

Figure 26: Deep learning based agent executes consecutive HPLC runs to find the best solvent gradient for peak separation of compounds.

MDQN loss function and other Deep Learning methods have been tested also in a commercial project within SIA ChromSword. As shown in 26. fig. in 3 consecutive runs, it is possible to find HPLC solvent gradient for the best peak separation of unknown compounds. These models are used in analytical chemistry and drug development applications.

6.3 Results of Exponential Triplet Loss Function

Results of Exponential Triplet loss L_{exp} for the class reidentification task are shown in 14. tab. All results were achieved with a grid search of all hyper-parameters. This is the reason that even the standard Triplet Loss L_{std} in many datasets achieved comparable results. All results have been classified using zero-shot models and the reidentification task, with the closest embedding of a sample to the same class center of mass. None of the classes and samples used for validation have been seen during the training time. In addition, the composite loss function has been used for both L_{std} and L_{exp} by adding Center Loss L_{cen} and L2-Softmax Classification Loss L_{cls}. Classification Loss was calculated only for the training dataset, because the test dataset contained different number and types of classes. In this research, a dataset for reidentification task was chosen as VGGFace2 [13] with 9000 classes, but the models also have been tested on classical image datasets like MNIST [55], Fassion-MNIST [130], EMNIST (Extended-MNIST) [15] and CIFAR10 [47]. For classical image data-sets, train and test sub-sets were re-divided by classes, so that the test set would not contain classes included in the train data-set. For all the datasets, 20% of classes with their samples were set aside for testing and 80% of different classes were left for training.

Table 14:

Comparison of zero-shot accuracy on test dataset for different loss functions (Triplet Loss L_{std}, Exponential Triplet Loss L_{exp}, Center Loss L_{cen}, Classification Loss L_{cls}

Loss / Acc.	MNIST	FMNIST	EMINST	CIFAR10	Simpsons	VGGFace2
L_{std}	**99.6**	91.4	82.0	56.2	91.0	77.4
$L_{std} + L_{cls}$	**99.6**	92.1	85.0	79.8	91.2	76.3
$L_{std} + L_{cen}$	97.5	71.5	61.7	52.1	90.9	76.4
$L_{std} + L_{cen} + L_{cls}$	97.7	82.0	70.9	62.8	91.2	78.6
L_{exp}	**99.6**	92.7	82.7	85.7	91.5	85.0
$L_{exp} + L_{cls}$	**99.6**	**93.1**	85.2	87.2	90.9	84.1
$L_{exp} + L_{cen}$	**99.6**	**93.1**	85.7	85.3	**92.1**	84.0
$L_{exp} + L_{cen} + L_{cls}$	**99.6**	**93.1**	**86.0**	**87.3**	91.7	**85.7**

Additionally, The Simpsons dataset provided by Kaggle was also analyzed for the reidentification task. The results for zero-shot learning depicted in 27. fig. showed good separation in 3D between visually different characters. As seen in the example below, the model achieved clustering of class and visual features only by applying linear PCA.

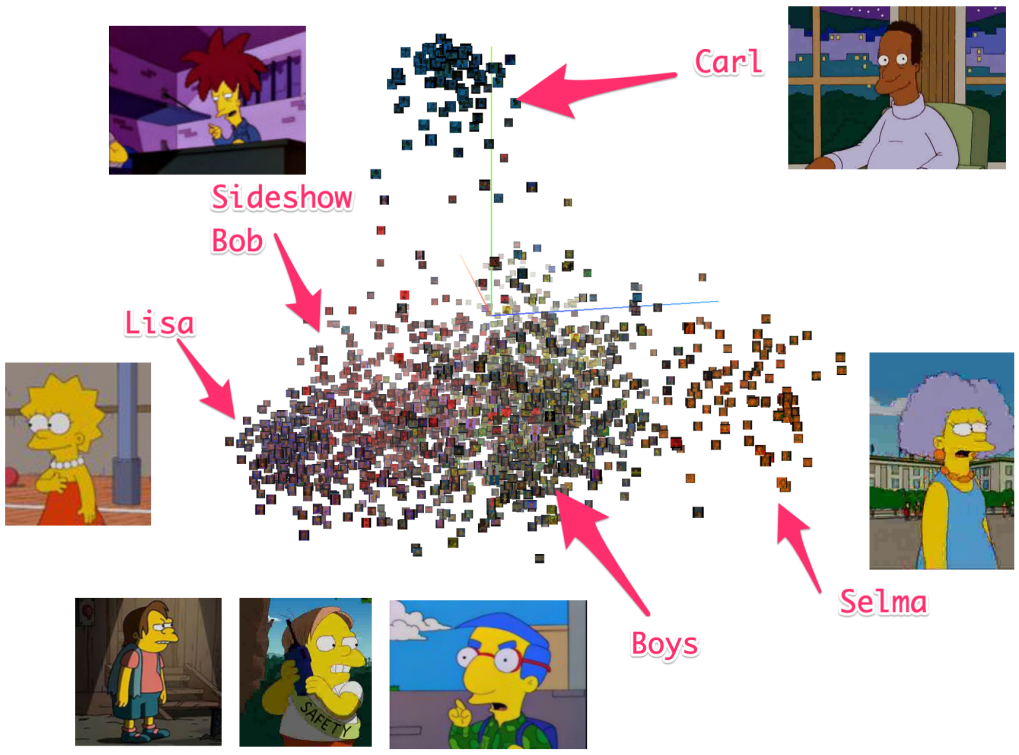

Figure 27: Visualization of PCA of Simpsons test dataset that Exponential Triplet loss-based model has not used for training.

6.4 Practical applications

Novel loss functions and models described in the thesis can be applied to many applications, some of these applications have already been implemented, others have not yet been tested. Practical applications for each of the novel loss functions and models are listed below.
Practical applications of UNet-RNN-Skip:

1. Optimization of VI algorithm for faster convergence of policy in path planning task. This application has been tested within the thesis. This model can achieve multiple orders of magnitude faster convergence than VI algorithm and is more scalable than VI algorithm on larger maps even though it has been trained only on smaller maps.

2. Optimization of VI algorithm for RL tasks. Policy can be calculated also for other tasks such as sequential planning where instead of converging policy to the shortest path in the map states can represent observations and gradient of value in the graph of actions can represent decisions.

3. Style transfer tasks for video data [40] [120]. Novel model presented in the thesis can be used to colorize black and white movies, add noise to make it look authentic, convert movies to look like cartoons.

4. Style transfer tasks for audio data [16] [1]. Novel model presented in the thesis can be used to clone voices or generate artificial voices using recorded source voice as an input. Model can be used with 1D audio signal or 2D spectrographs that later are converted back to audible signal using Griffin-Lim algorithm.

5. Video denoising tasks [20] [105]. Novel model can be used to remove noise and film degradation artifacts as well it can be used to improve compression of video by reducing complexity in regions that human observer would not pay attention.

6. Audio denoising tasks [58] [131]. This task has been successfully applied in commercial solutions within SIA Asya. Models are used in real-time to remove background noises in natural environments and leaving just signal of voice. Because of this model commercial implementation of asya.ai mobile application is able to provide conversational analysis in noisy environments such as coffee shops or offices.

Practical applications of MDQN Loss function:

1. Training Q-Value function based policy for a wide variety of RL tasks like simulations [42], self-driving [134], virtual assistants [50] and robot control [77]. In the thesis novel MDQN loss function has been tested using PLE computer games. MDQN loss function is especially efficient when using with low dimensional state inputs or using Deep Metric Learning and low dimensional state embeddings.

2. Real-time optimization of solvent gradients for HPLC in analytical chemistry [26]. It has been successfully tested in commercial solutions within SIA ChromSword. Model is capable of finding the gradient of solvents for compound separation in 2 hours when before manual sequences of experiments could take multiple days.

Practical applications of Exponential Triplet Loss function:

1. Pre-training of encoder using deep metric learning of embeddings for a wide variety of tasks like RL, RNN, classification and regression [129]. Pre-training encoder using Exponential Triplet Loss can speed up the training process of the main task. The precondition for this method is that a dataset used for training must be labelled or similarity measure between samples must be included. But this measure also can be approximated using unsupervised learning methods like VAE (Variational Auto Encoders) [19].

2. Re-identification task for images or biometric data [23] [5]. The novel loss function has been tested within the thesis in the context of face and image re-identification. It has also been successfully applied in commercial solution within SIA Asya for voice re-identification task. Asya mobile application executes the re-identification of multiple speakers using voice biometric data for natural conversations in real-time. It could also be applied for e-commerce solutions for re-identification of different products by their visual similarity.

7 FUTURE RESEARCH

Future research of the novel loss functions, their attributes, and shape can be based on the findings presented in the thesis.
Research in Deep Q-Learning and Reinforcement Learning is very tied to loss function as it is the main and unchaining factor in an unpredictable environment. One extension to the loss function of Deep Q-Learning would be to add a loss that would emulate curiosity and exploration of environment. Loss function that emulates curiosity should not be based on rewards of the environment itself, but instead unexpected and novel state information should generate an intrinsic reward. At the same time, random events that are not linked to the actions of an agent should not be included in this intrinsic reward. Some work has been done in this direction, but it has not been fully solved [12] [11].
Another direction would be to add to the model multiple read and write heads to train the model to use auxiliary memory tables in the context of Deep Reinforcement Learning. Memory tables should contain embeddings of states, previous states, and rewards. These models would learn to store and use the facts observed in the environment and use them to maximize reward even though these facts have never been observed during the training process, similarly to how zero-shot learning works. The model would learn the algorithm that can solve the environment, not the patterns of an environment. To train such models, a composite loss function that includes the regularization of heads could be added. Recent work has been done also in this direction, but also it has not yet been solved for complex environments [125] [81] [51].
The loss functions needed to map the similarity of embeddings of read and write heads for auxiliary memory tables and curiosity models are similar to those used in Deep Metric Learning like Triplet Loss or Contrastive Loss.
Deep Metric Learning in the context of Zero-Shot, One-Shot k-Shot learning also has not yet been solved.
Extension of the research presented in the thesis might be adding KL (KullbackLeibler) divergence or other probability density similarity functions and adding prior distributions. Usually KL is used together with a Gaussian distribution with learnable mean and standard deviation, but the loss function could enable the model to learn also other distributions. In addition, it might be useful to add the reconstruction loss generated from VAE and the adversarial loss generated from GAN.

Recently, novel loss functions like Margin Loss in CapsNet (Capsule Networks) [89] and Focal Loss [61] have been used also for classification problems instead of the commonly used cross-entropy loss function that comes from established information theory. Both of these functions have different properties of shape of the function that leads to better convergence. There could be even better versions of loss functions for classification task and other common machine learning tasks, and the shape of the function might be an important factor.

8 CONCLUSIONS

The thesis has proposed and evaluated novel loss functions, model architectures, and training algorithms. It has presented research findings of the importance of the shape of functions for deep learning methods.
Contributions of this work include UNet-RNN-Skip model, OccupancyMap-Generator algorithm, MDQN Loss function, Q-Value maps, Exponential Triplet Loss function, Unit-Range latent space normalization function, Unit-Bounce latent space normalization function and other methods that have been published in the scientific literature produced by the author and attached to the Thesis.
The experimental results show that the proposed methods achieve better results than the established deep learning methods.
For the face re-identification task and for the deep metric learning task on VGGFace2 dataset, Exponential Triplet Loss function reached state-of-the-art results of 85.7% accuracy using zero-shot setting. The exponential Triplet Loss function also converges faster than the conventional Triplet Loss function with common composite loss function addition. Unit-Range normalization function and Unit-Bounce normalization function achieve better utilization of the embedding space than L2 normalization function and have similar properties to the cosine distance in Euclidean space.
For the reinforcement learning task in computer game environments, MDQN loss function achieves higher scores than DDQN and DQN loss functions. It also provides functionality to construct Q-Value map that exposes the model policy as a white box to understand the better decision-making process using a visual representation of each state. The results also showed the need of repeated experiments with the same set of hyper-parameters of MDQN and other loss functions at least 20 times to reduce the effect of random weight

initialization in highly random environments like PLE. MDQN loss function research also included a survey of state-of-the-art methods at the time of publishing and analysis of scientific literature regarding Deep Q-Learning.
For Value function modelling, UNet-RNN-Skip execution speed is on the order of magnitude greater than the classical Value function used in VI. It also can have the same policy outcome for 99.8% of the cases and can be trained on 32x32 maps, but then applied to larger maps like 256x256. OccupancyMapGenerator can be applied successfully to generate synthetic datasets of occupancy maps. These datasets can be used for tasks related to VI algorithm as well as for other tasks such as SLAM.
All the theses listed in 1.2. sec. have been confirmed.
Novel MDQN and Exponential Triplet Loss functions have been successfully applied in commercial products for analytical chemistry task at SIA ChromSword and for voice and face reidentification task at SIA Asya. UNet-RNN-Skip also has been successfully applied in a commercial product for noise reduction of audio signal at SIA Asya.
The contributions of this research can be used in different applications that have been described in 6.4. sec. Applications include zero-shot tasks not only for biometric reidentification, but also finding similar products using photos. Applications also include path planning, analytical chemistry, automatic speech recognition, reinforcement learning, and robot control. The work emphasizes the efficiency of unconventional loss functions and approaches in Deep Learning that have been developed and shaped using empirical methods.

ACKNOWLEDGEMENTS

Research has been completed with a support from RTU (Riga Technical University) IKSA Research Lab and RTU HPC (High-Performance Computing Center). RTU HPC provided 12 nVidia K40 GPUs and 8 nVidia V100 GPUs. Special thanks to RTU HPC manager Lauris Cikovskis and RTU IKSA Research Lab manager, RTU dean Agris Nikitenko.

BIBLOGRAPHY

[1] Sercan Ö. Arik et al. “Neural Voice Cloning with a Few Samples”. In: *ArXiv* abs/1802.06006 (2018).

[2] Kai Arulkumaran et al. “A Brief Survey of Deep Reinforcement Learning”. In: *ArXiv* abs/1708.05866 (2017).

[3] Jimmy Ba, J. Kiros, and Geoffrey E. Hinton. “Layer Normalization”. In: *ArXiv* abs/1607.06450 (2016).

[4] Mohammad Babaeizadeh et al. “GA3C: GPU-Based A3C for Deep Reinforcement Learning”. In: *CoRR* abs/1611.06256 (2016). URL: `http://arxiv.org/abs/1611.06256`.

[5] H. Bredin. “TristouNet: Triplet Loss for Speaker Turn Embedding”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017), pp. 5430–5434.

[6] Greg Brockman et al. “OpenAI Gym”. In: *ArXiv* abs/1606.01540 (2016).

[7] J. Bromley et al. “Signature Verification Using a "Siamese" Time Delay Neural Network”. In: *Int. J. Pattern Recognit. Artif. Intell.* 1993.

[8] Jane Bromley et al. “Signature Verification Using A "Siamese" Time Delay Neural Network”. In: *IJPRAI* 7.4 (1993), pp. 669–688. DOI: `10.1142/S0218001493000339`. URL: `https://doi.org/10.1142/S0218001493000339`.

[9] Edward De Brouwer et al. “GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series”. In: *ArXiv* abs/1905.12374 (2019).

[10] T. Brown et al. “Language Models Are Few-Shot Learners”. In: *ArXiv* abs/2005.14165 (2020).

[11] Yuri Burda et al. “Exploration by Random Network Distillation”. In: *ArXiv* abs/1810.12894 (2019).

[12] Yuri Burda et al. “Large-Scale Study of Curiosity-Driven Learning”. In: *ArXiv* abs/1808.04355 (2019).

[13] Qiong Cao et al. “VGGFace2: A Dataset for Recognising Faces across Pose and Age”. In: *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)* (2017), pp. 67–74.

[14] Sumit Chopra, Raia Hadsell, and Yann LeCun. “Learning a Similarity Metric Discriminatively, with Application to Face Verification”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. 2005, pp. 539–546. DOI: 10.1109/CVPR.2005.202. URL: https://doi.org/10.1109/CVPR.2005.202.

[15] Gregory Cohen et al. “EMNIST: An Extension of MNIST to Handwritten Letters”. In: *ArXiv* abs/1702.05373 (2017).

[16] J. Cong et al. “Data Efficient Voice Cloning from Noisy Samples with Domain Adversarial Training”. In: *ArXiv* abs/2008.04265 (2020).

[17] Haowen Deng, Tolga Birdal, and Slobodan Ilic. “PPFNet: Global Context Aware Local Features for Robust 3D Point Matching”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 195–205.

[18] E. Dijkstra. “A Note on Two Problems in Connexion with Graphs”. In: *Numerische Mathematik* 1 (1959), pp. 269–271.

[19] E. Dupont. “Learning Disentangled Joint Continuous and Discrete Representations”. In: *NeurIPS*. 2018.

[20] T. Ehret et al. “Model-Blind Video Denoising via Frame-to-Frame Training”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 11361–11370.

[21] Lasse Espeholt et al. “IMPALA: Scalable Distributed Deep-Rl with Importance Weighted Actor-Learner Architectures”. In: *ArXiv* abs/1802.01561 (2018).

[22] Chelsea Finn, S. Levine, and P. Abbeel. “Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization”. In: *ArXiv* abs/1603.00448 (2016).

[23] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “FaceNet: A Unified Embedding for Face Recognition and Clustering”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682. URL: https://doi.org/10.1109/CVPR.2015.7298682.

[24] S. Galushko et al. “ChromSword: Software for Method Development in Liquid Chromatography”. In: 2018.

[25] Yuan Gao and Dorota Glowacka. "Deep Gate Recurrent Neural Network". In: *ArXiv* abs/1604.02910 (2016).

[26] Tarun Gogineni et al. "TorsionNet: A Reinforcement Learning Approach to Sequential Conformer Search". In: *ArXiv* abs/2006.07078 (2020).

[27] Jacob Goldberger et al. "Neighbourhood Components Analysis". In: *Advances in Neural Information Processing Systems*. Ed. by L. Saul, Y. Weiss, and L. Bottou. Vol. 17. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf.

[28] Klaus Greff et al. "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (Oct. 2017), pp. 2222–2232. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2016.2582924. arXiv: 1503.04069. URL: http://arxiv.org/abs/1503.04069 (visited on 04/25/2019).

[29] Hado V. Hasselt. "Double Q-Learning". In: *Advances in Neural Information Processing Systems 23*. Ed. by J. D. Lafferty et al. Curran Associates, Inc., 2010, pp. 2613–2621. URL: http://papers.nips.cc/paper/3964-double-q-learning.pdf.

[30] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning". In: *CoRR* abs/1509.06461 (2015). URL: http://arxiv.org/abs/1509.06461.

[31] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

[32] Kaiming He et al. "Identity Mappings in Deep Residual Networks". In: *ArXiv* abs/1603.05027 (2016).

[33] Alexander Hermans, Lucas Beyer, and Bastian Leibe. "In Defense of the Triplet Loss for Person Re-Identification". In: *ArXiv* abs/1703.07737 (2017).

[34] J. Hershey et al. "Deep Clustering: Discriminative Embeddings for Segmentation and Separation". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), pp. 31–35.

[35] Matteo Hessel et al. “Rainbow: Combining Improvements in Deep Reinforcement Learning”. In: *ArXiv* abs/1710.02298 (2018).

[36] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9 (1997), pp. 1735–1780.

[37] C. Huang, Chen Change Loy, and X. Tang. “Local Similarity-Aware Deep Feature Embedding”. In: *NIPS*. 2016.

[38] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. “Densely Connected Convolutional Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2261–2269.

[39] Huimin Huang et al. “UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), pp. 1055–1059.

[40] Satoshi Iizuka and Edgar Simo-Serra. “DeepRemaster: Temporal Source-Reference Attention Networks for Comprehensive Video Enhancement”. In: *ArXiv* abs/2009.08692 (2019).

[41] Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. “An Empirical Exploration of Recurrent Network Architectures”. In: *ICML*. 2015.

[42] A. Kiani, Chris Wang, and Angela Xu. “Sepsis World Model: A MIMIC-Based OpenAI Gym "World Model" Simulator for Sepsis Treatment”. In: *ArXiv* abs/1912.07127 (2019).

[43] B. Kitchenham et al. “Systematic Literature Reviews in Software Engineering - A Systematic Literature Review”. In: *Inf. Softw. Technol.* 51 (2009), pp. 7–15.

[44] Gregory R. Koch. “Siamese Neural Networks for One-Shot Image Recognition”. In: 2015.

[45] A. Kolesnikov et al. “Big Transfer (BiT): General Visual Representation Learning”. In: *arXiv: Computer Vision and Pattern Recognition* (2019).

[46] Martin Köstinger et al. “Large Scale Metric Learning from Equivalence Constraints”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 2288–2295.

[47] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: 2009.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Communications of the ACM* 60.6 (May 24, 2017), pp. 84–90. ISSN: 00010782. DOI: 10.1145/3065386. URL: http://dl.acm.org/citation.cfm?doid=3098997.3065386 (visited on 04/25/2019).

[49] David Krueger et al. “Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations”. In: *ArXiv* abs/1606.01305 (2017).

[50] Katya Kudashkina, P. Pilarski, and R. Sutton. “Document-Editing Assistants and Model-Based Reinforcement Learning as a Path to Conversational AI”. In: *ArXiv* abs/2008.12095 (2020).

[51] Guillaume Lample et al. “Large Memory Layers with Product Keys”. In: *ArXiv* abs/1907.05242 (2019).

[52] M. Law, N. Thome, and M. Cord. “Quadruplet-Wise Image Similarity Learning”. In: *2013 IEEE International Conference on Computer Vision* (2013), pp. 249–256.

[53] M. Law, R. Urtasun, and R. Zemel. “Deep Spectral Clustering Learning”. In: *ICML*. 2017.

[54] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. “A Simple Way to Initialize Recurrent Networks of Rectified Linear Units”. In: *ArXiv* abs/1504.00941 (2015).

[55] Yann LeCun and Corinna Cortes. “MNIST Handwritten Digit Database”. In: (2010). URL: http://yann.lecun.com/exdb/mnist/ (visited on 01/14/2016).

[56] Yann LeCun et al. “Comparison of Learning Algorithms for Handwritten Digit Recognition”. In: 1995.

[57] H. Lee et al. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”. In: *ICML ’09*. 2009.

[58] J. Lee et al. “Dynamic Noise Embedding: Noise Aware Training and Adaptation for Speech Enhancement”. In: 2020.

[59] Chao Li et al. “Deep Speaker: An End-to-End Neural Speaker Embedding System”. In: *CoRR* abs/1705.02304 (2017). arXiv: 1705.02304. URL: http://arxiv.org/abs/1705.02304.

[60] Timothy P. Lillicrap et al. “Continuous Control with Deep Reinforcement Learning”. In: *CoRR* abs/1509.02971 (2015). URL: http://arxiv.org/abs/1509.02971.

[61] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2999–3007.

[62] Teng Long et al. “Zero-Shot Learning via Discriminative Representation Extraction”. In: *Pattern Recognition Letters* 109 (2018), pp. 27–34.

[63] Scott Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *NIPS*. 2017.

[64] R. Manmatha et al. “Sampling Matters in Deep Embedding Learning”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2859–2867.

[65] Michael Phi. *Illustrated Guide to LSTMs and GRUs: A Step by Step Explanation*. Jan. 6, 2020. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[66] Erik G. Miller, Nicholas E. Matsakis, and Paul A. Viola. “Learning from One Example through Shared Densities on Transforms”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)* 1 (2000), 464–471 vol.1.

[67] Volodymyr Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *ICML* 48 (2016), pp. 1928–1937. URL: http://arxiv.org/abs/1602.01783.

[68] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). URL: http://arxiv.org/abs/1312.5602.

[69] Igor Mordatch. “Concept Learning with Energy-Based Models”. In: *ICLR*. 2018.

[70] Yair Movshovitz-Attias et al. “No Fuss Distance Metric Learning Using Proxies”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 360–368.

[71] “N-Shot Learning: Learning More with Less Data”. In: (). URL: https://blog.floydhub.com/n-shot-learning/.

[72] D. Neil, M. Pfeiffer, and Shih-Chii Liu. “Phased LSTM: Accelerating Recurrent Network Training for Long or Event-Based Sequences”. In: *NIPS*. 2016.

[73] Binh X. Nguyen et al. “Deep Metric Learning Meets Deep Clustering: An Novel Unsupervised Approach for Feature Embedding”. In: *ArXiv* abs/2009.04091 (2020).

[74] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. “Learning Deconvolution Network for Semantic Segmentation”. In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1520–1528.

[75] “Non-Zero Initial States for Recurrent Neural Networks - R2RT”. In: (). URL: https://r2rt.com/non-zero-initial-states-for-recurrent-neural-networks.html.

[76] OpenAI et al. “Solving Rubik’s Cube with a Robot Hand”. In: *ArXiv* abs/1910.07113 (2019).

[77] OpenAI et al. “Solving Rubik’s Cube with a Robot Hand”. In: *ArXiv* abs/1910.07113 (2019).

[78] D. Park et al. “Improved Noisy Student Training for Automatic Speech Recognition”. In: *ArXiv* abs/2005.09629 (2020).

[79] Adam Paszke et al. “Automatic Differentiation in PyTorch”. In: (2017).

[80] K. Petersen et al. “Systematic Mapping Studies in Software Engineering”. In: *EASE*. 2008.

[81] Alexander Pritzel et al. “Neural Episodic Control”. In: *ArXiv* abs/1703.01988 (2017).

[82] Qi Qi et al. “A Simple and Effective Framework for Pairwise Deep Metric Learning”. In: *Computer Vision ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 375–391. ISBN: 978-3-030-58583-9.

[83] Hubert Ramsauer et al. “Hopfield Networks Is All You Need”. In: *ArXiv* abs/2008.02217 (2020).

[84] Rajeev Ranjan, Carlos D. Castillo, and Rama Chellappa. “L2-Constrained Softmax Loss for Discriminative Face Verification”. In: *CoRR* abs/1703.09507 (2017). arXiv: 1703.09507. URL: http://arxiv.org/abs/1703.09507.

[85] Mirco Ravanelli, Titouan Parcollet, and Yoshua Bengio. “The Pytorch-Kaldi Speech Recognition Toolkit”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), pp. 6465–6469.

[86] Oren Rippel et al. “Metric Learning with Adaptive Density Discrimination”. In: *ICLR* abs/1511.05939 (2016).

[87] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *MICCAI*. 2015.

[88] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, Dec. 2002. ISBN: 0-13-790395-2. URL: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0137903952.

[89] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. “Dynamic Routing between Capsules”. In: *ArXiv* abs/1710.09829 (2017).

[90] Tom Schaul et al. “Prioritized Experience Replay”. In: *CoRR* abs/1511.05952 (2015). URL: http://arxiv.org/abs/1511.05952.

[91] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *ArXiv* abs/1707.06347 (2017).

[92] John Schulman et al. “Trust Region Policy Optimization”. In: *ICML*. 2015.

[93] Ramprasaath R. Selvaraju et al. “Grad-CAM: Why Did You Say That? Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *CoRR* abs/1610.02391 (2016). URL: http://arxiv.org/abs/1610.02391.

[94] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. “Recurrent Dropout without Memory Loss”. In: *COLING*. 2016.

[95] M. Shoeybi et al. “Megatron-Lm: Training Multi-Billion Parameter Language Models Using Model Parallelism”. In: *ArXiv* abs/1909.08053 (2019).

[96] Ari Silburt et al. “Lunar Crater Identification via Deep Learning”. In: *Icarus* 317 (2019), pp. 27–38.

[97] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2015).

[98] Kihyuk Sohn. “Improved Deep Metric Learning with Multi-Class n-Pair Loss Objective”. In: *NIPS*. 2016.

[99] Hyun Oh Song et al. “Deep Metric Learning via Facility Location”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2206–2214.

[100] Hyun Oh Song et al. “Deep Metric Learning via Lifted Structured Feature Embedding”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 4004–4012.

[101] Hyun Oh Song et al. “Learnable Structured Clustering Framework for Deep Metric Learning”. In: *ArXiv* abs/1612.01213 (2016).

[102] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning”. In: *AAAI*. 2016.

[103] Haoran Tang et al. “#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning”. In: *ArXiv* abs/1611.04717 (2017).

[104] Norman Tasfi. “PyGame Learning Environment”. In: *GitHub repository* (2016). URL: `https://github.com/ntasfi/PyGame-Learning-Environment`.

[105] Matias Tassano, J. Delon, and T. Veit. “FastDVDnet: Towards Real-Time Deep Video Denoising without Flow Estimation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 1351–1360.

[106] “Tips for Training Recurrent Neural Networks”. In: (). URL: `https://danijar.com/tips-for-training-recurrent-neural-networks/`.

[107] Hugo Touvron et al. “Fixing the Train-Test Resolution Discrepancy”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[108] “Triplet Loss and Online Triplet Mining in TensorFlow | Olivier Moindrot Blog”. In: (). URL: `https://omoindrot.github.io/triplet-loss`.

[109] E. Urtans and Agris Nikitenko. “Survey of Deep Q-Network Variants in PyGame Learning Environment”. In: *ICDLT ’18*. 2018.

[110] E. Urtans, Agris Nikitenko, and Valters Vecins. “Exponential Triplet Loss”. In: *Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis* (2020).

[111] E. Urtans and Valters Vecins. “Value Iteration Solver Networks”. In: *2020 3rd International Conference on Intelligent Autonomous Systems (ICoIAS)* (2020), pp. 8–13.

[112] Evalds Urtans and Ariel Tabaks. *Asya: Mindful Verbal Communication Using Deep Learning*. 2020. arXiv: `2008.08965 [eess.AS]`.

[113] E. Ustinova and V. Lempitsky. “Learning Deep Embeddings with Histogram Loss”. In: *NIPS*. 2016.

[114] R. Vaillant, C. Monrocq, and Y. Le Cun. “Original Approach for the Localisation of Objects in Images”. In: *IEE Proceedings - Vision, Image and Signal Processing* 141.4 (1994), pp. 245–250.

[115] Athanasios Voulodimos et al. “Deep Learning for Computer Vision: A Brief Review”. In: *Computational Intelligence and Neuroscience* 2018 (2018).

[116] Chanchin Wang, Xue Zhang, and Xipeng Lan. “How to Train Triplet Networks with 100K Identities?” In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)* (2017), pp. 1907–1915.

[117] J. Wang et al. “Deep Metric Learning with Angular Loss”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2612–2620.

[118] Jinjiang Wang et al. “Deep Learning for Smart Manufacturing: Methods and Applications”. In: *Journal of Manufacturing Systems* 48 (2018), pp. 144–156.

[119] X. Wang et al. “Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 5017–5025.

[120] Xinrui Wang and Jinze Yu. “Learning to Cartoonize Using White-Box Cartoon Representations”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 8087–8096.

[121] Xinshao Wang et al. “Ranked List Loss for Deep Metric Learning”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 5202–5211.

[122] Ziyu Wang, Nando de Freitas, and Marc Lanctot. “Dueling Network Architectures for Deep Reinforcement Learning”. In: *CoRR* abs/1511.06581 (2015). URL: http://arxiv.org/abs/1511.06581.

[123] Ziyu Wang, Nando de Freitas, and Marc Lanctot. “Dueling Network Architectures for Deep Reinforcement Learning”. In: *CoRR* abs/1511.06581 (2015). URL: http://arxiv.org/abs/1511.06581.

[124] Ziyu Wang et al. “Sample Efficient Actor-Critic with Experience Replay”. In: *ArXiv* abs/1611.01224 (2017).

[125] Greg Wayne et al. “Unsupervised Predictive Memory in a Goal-Directed Agent”. In: *ArXiv* abs/1803.10760 (2018).

[126] “We Analyzed 16,625 Papers to Figure out Where AI Is Headed next | MIT Technology Review”. In: (). URL: https://www.technologyreview.com/2019/01/25/1436/we-analyzed-16625-papers-to-figure-out-where-ai-is-headed-next/.

[127] Kilian Q. Weinberger and L. Saul. “Distance Metric Learning for Large Margin Nearest Neighbor Classification”. In: *NIPS*. 2005.

[128] Yandong Wen et al. “A Discriminative Feature Learning Approach for Deep Face Recognition”. In: *ECCV*. 2016.

[129] Cameron R. Wolfe and Keld T. Lundgaard. “E-Stitchup: Data Augmentation for Pre-Trained Embeddings”. In: 2019.

[130] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-Mnist: A Novel Image Dataset for Benchmarking Machine Learning Algorithms”. In: *ArXiv* abs/1708.07747 (2017).

[131] Yong Xu et al. “Dynamic Noise Aware Training for Speech Enhancement Based on Deep Neural Networks”. In: *INTERSPEECH*. 2014.

[132] Dong Yi, Zhen Lei, and S. Li. “Deep Metric Learning for Practical Person Re-Identification”. In: *ArXiv* abs/1407.4979 (2014).

[133] Y. Yuan, Kuiyuan Yang, and Chao Zhang. “Hard-Aware Deeply Cascaded Embedding”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 814–823.

[134] Q. Zhang and Tao Du. “Self-Driving Scale Car Trained by Deep Reinforcement Learning”. In: *ArXiv* abs/1909.03467 (2019).

[135] W. Zheng, S. Gong, and T. Xiang. “Reidentification by Relative Distance Comparison”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (2013), pp. 653–668.

[136] Zongwei Zhou et al. “UNet++: A Nested u-Net Architecture for Medical Image Segmentation”. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support : 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, held in conjunction with MICCAI 2018, Granada, Spain, S...* 11045 (2018), pp. 3–11.

APPENDIX A - Paper 1

Value Iteration Solver Networks

1st Evalds Urtans
Riga Technical University
Riga, Latvia
evalds.urtans@rtu.lv

2nd Valters Vecins
Riga Technical University
Riga, Latvia
valters.vecins@rtu.lv

***Abstract*—Value Iteration Algorithm is iterative and can't be parallelized. Computation time grows exponentially when the size of the input maps is increased. We propose *UNet-RNN-Skip* artificial neural network architecture that can be used to parallelize Value Iteration Algorithm results. The proposed model can solve Value Iteration problem in fewer iterations than the original algorithm and computation time increases by only a small amount when increasing the size of the input map. Fundamental *UNet-RNN-Skip* architecture can be used also to solve and parallelize other sequential problems. With this paper synthetic dataset of maps and generator has been published to enable further studies in mapping and path planning tasks.**

***Index Terms*—ResNet, ConvNet, RNN, Value Iteration Algorithm**

I. Introduction

Value Iteration Algorithm (VI) is widely used to generate navigation maps and policies for a wide range of problems where each state can have different value [1].

For navigation task policy for movement is calculated to the direction of the gradient of adjacent grid cell values within a discretized map. Value Iteration Algorithm is an iterative process where the values of each grid cell depend on the values of the previous iteration of adjacent grid cells. Within Value Iteration Algorithm given in (1) for each state s value $V(s)$ is calculated by choosing action a that maximizes sum of reward with given action R_a, multiplied by transition probability P_a and added adjacent state values $V(s')$ multiplied by γ discount factor. Formally state s consists of the state that includes all gird cells in the map, but for a simplified explanation, we can assume that each grid cell has its own state.

Value Iteration Algorithm ensures optimal policy within fully observable environment Fig. 1. In this case, it ensures that a positive terminal state is reachable from every cell in the map by following a policy that guides by the closest path to the target.

Using Value Iteration Algorithm for path planning in real-time is often limited to its performance as it becomes exponentially slower as the map becomes larger.

$$V_{i+1}(s) := \max_a \{\sum_{s'} P_a(s,s')(R_a(s,s') + \gamma V_i(s'))\} \quad (1)$$

An alternative approach is to use heuristic-based approaches for path planning such as Dijkstra or A* algorithms [1]. These are much faster than VI and are widely used in simulations and computer games, but they do not ensure optimal policy.

Another approach is to utilize the latest research in Deep Artificial Neural Networks hence we propose Value Iteration Solver Network model. Currently, similar models have been applied to a wide range of problems starting with image classification, image segmentation and policy selection of agent in the reinforcement learning.

Architectures of these models are different depending on the task, but basic concepts are common in between them. Convolutional artificial neural network architectures that we are proposing in this research are based upon AlexNet [2], VGG [3], InceptionNet [4], ResNet [5], DenseNet [6] and UNet [7].

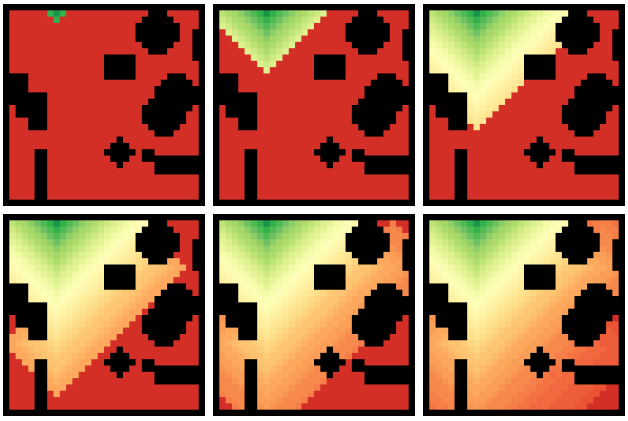

Fig. 1. Visualization of Value Iteration Algorithm's consecutive iterations. After convergence, it is possible to derive optimal policy from every state to reach the terminal state with highest cumulative reward.

II. Related work

Value iteration algorithm is a variation of the Markov decision process (MDP) for finding the optimal policy in discrete state-action space.

Recently Value Iteration Networks has been proposed as a novel neural model-based algorithm that focuses on mimicking the behaviour of Value Iteration Algorithm by iterating over values multiple times in the inner loop of convolutional architecture [8]. Even though the inner workings of such model have high research value, the results produced by this method are not robust enough for practical use. Other research has been done in solving Value Iteration problem by using reinforcement learning. "Second Order Value Iteration in Reinforcement Learning" proposes using Newton-Raphson method

for second-value iteration algorithm for faster convergence to almost optimal values [9].

Also, there have been other developments in this field of research like "Value Iteration Networks on Multiple Levels of Abstraction" that extends the work of original "Value Iteration Networks" by processing input in multiple levels of abstraction. With multiple levels of abstraction, they increase the success value metric for larger maps. [10]

III. Methodology

Within this research, we developed a novel artificial neural network architecture that can be used for different problems, and to achieve the same results as Value Iteration Algorithm for map navigation task. These models are based on ResNet and UNet architectures as well as on Recurrent Neural Networks.

A. UNet variant

UNet architecture was first introduced to solve biomedical image segmentation [7].

Solving the value iteration algorithm task is similar to the segmentation task, a network output should retain a lot of features from the input image like walls and obstacles in a map. Solving value iteration with convolutional neural networks using UNet architecture helps the network to learn how value iteration values propagate through the map because all of the information doesn't have to be encoded in the latent vector. UNet skip connections allow passing different abstraction latent representations from encoder to decoder Fig. 2.

While using ResNet or UNet architectures it is possible to concatenate or add the values of skip connections [11]. In our network we are using addition operation, so all of the information from the encoder is used by the decoder and gradient from error through back-propagation is distributed evenly. Concatenating the values would cause some of the values to be used more than others by the network. This model is similar to the denoising auto-encoder task, but instead of encoding features, we use it as the single iteration filter to get the map of state values and policies.

B. UNet-RNN variant

With UNet-RNN variant we introduced recurrent neural network cell and slightly changed the definition of the task Fig. 3. With recurrent models, we are modelling value iterations not within a single step, but within multiple iterations where an output of a previous iteration is fed into the next iteration. In fact the model learns to include in a single iteration multiple steps of the value iteration algorithm thus reducing, even more, the time needed to generate the value map. For recurrent part, we tested different versions of LSTM and GRU and found out that single-layer LSTM had the best performance [12].

With the use of the recurrent layer, models task is simplified because the model can do multiple iterations on the same map. In theory, this allows the model to produce better value predictions for map places with narrow corridors or obstacles.

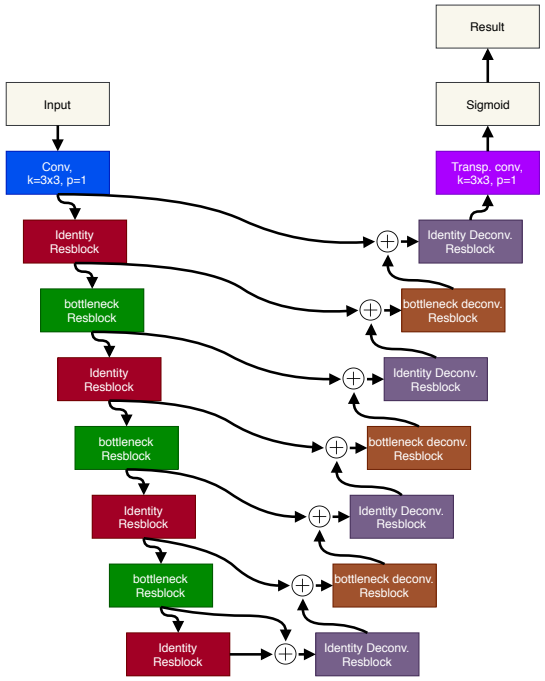

Fig. 2. Residual UNet architecture. Colors denote different building blocks used for model.

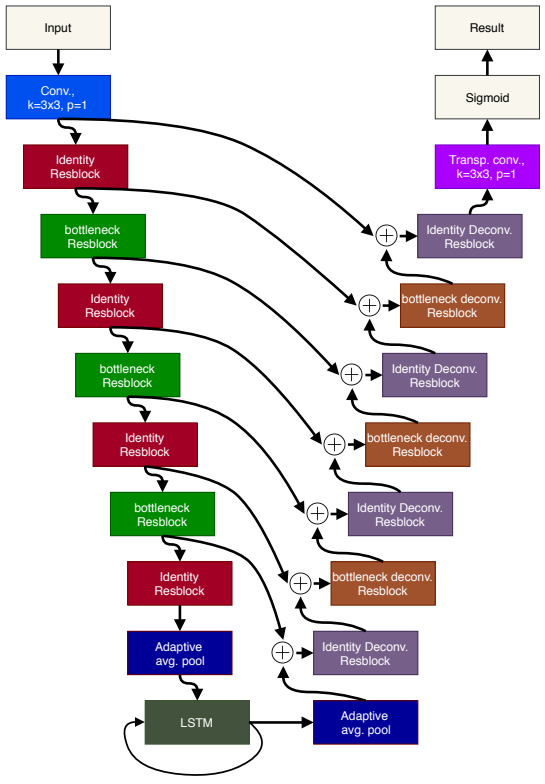

Fig. 3. UNet-RNN architecture. Colors denote different building blocks used for model.

C. UNet-RNN-Skip variant

With UNet-RNN-Skip variant we introduced a novel model architecture that is built on UNet-RNN. It is similar to the approach used in DenseNet architecture [6], but applied to UNet-RNN. In this model's architecture, we propose to add skip connections within the encoder and the decoder part itself Fig. 4. For the encoder and the decoder, we use 2 skip connections, each going over 3 residual blocks within the same part of the model. These skip connections, in theory, allows the model to maintain details of the map at different scales and different feature abstractions. For joining skip connections we used addition operation as before.

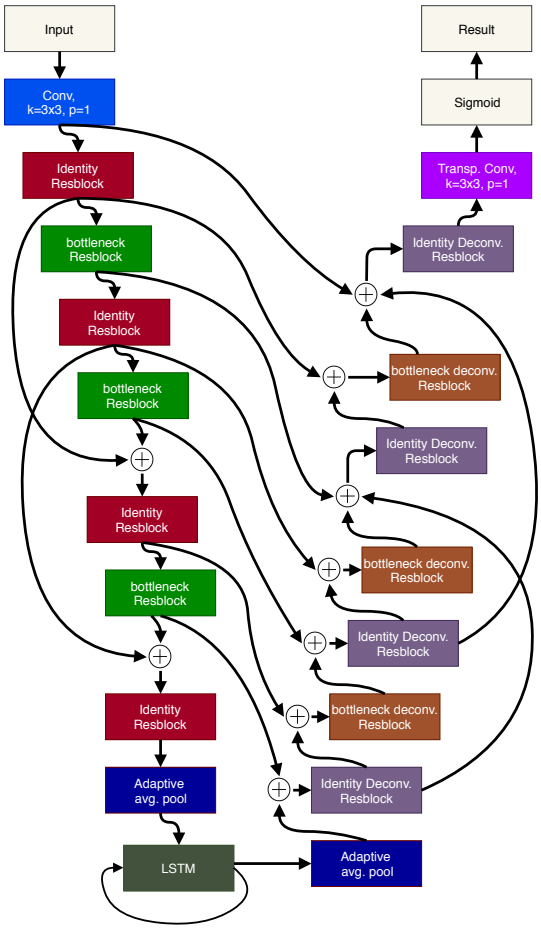

Fig. 4. UNet-RNN-Skip architecture. Colors denote different building blocks used for model.

IV. Experiments

A. Data set

While there are some available synthetic data-set generation tools available [8], we created our grid map generator of varying complexity. The grid map generator can generate maps of different sizes and different coverage of obstacles in the map. It is possible to choose between different obstacle types and map types, for example, rectangular obstacles or obstacles generated from the maze patterns Fig. 5. Source code is available at https://gitlab.com/VVecins/OccupancyMapGenerator.git. On synthetically generated grid maps we applied the value iteration algorithm to create the ground truth data-set for the training of the model. For the value iteration algorithm generator, changeable parameters include the map size, the discount value, allowed movement types and environment type (deterministic or stochastic). In this research, iterations were saved when the delta for values between iterations was higher than predefined constant (0.1). Not saving all iterations was for training RNN model to solve the multiple value iterations in one iteration. Source code available at https://gitlab.com/VVecins/ValueIterationGridmap.git.

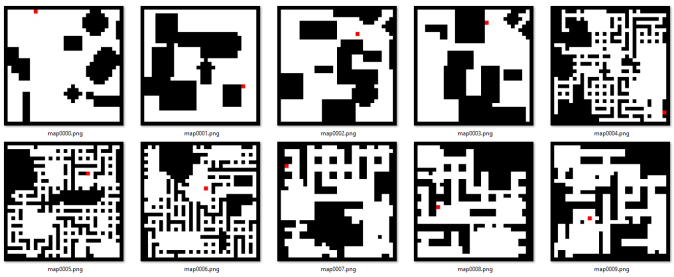

Fig. 5. Examples of synthetic maps generated by OccupancyMapGenerator.

B. Implementation details

All of the source code of our models are open-source and have been implemented using pytorch library. With pytorch it is possible to execute UNet-RNN-Skip mostly in a parallel manner as we processed all data of encoder and decoder parts together. Some parts of the training process of the model like our metrics "success score" and "success rate" are also made to be executed in a parallel manner using multiprocessing [13].

C. Metrics

For the loss function, we were using MSE loss and modified MSE loss function. Modified MSE loss function consists of average MSE from output and error for the grid cell value of the positive terminal state. Modified MSE loss function was introduced because by using regular MSE models tend to get good values by copying obstacles and then reach a plateau when learning gradient of values. An added loss for the terminal state decreases iteration count for when the model to correctly predict gradient values Fig. 6.

We introduce two metrics for evaluating the model performance, success rate metric Fig. 7 and score metric. By calculating from how many states we can reach the positive terminal state following the gradient of adjacent values of the grid, we calculate success rate and by calculating the sum of all transitions needed to reach the positive terminal state from every state of the map, we calculate score metric. If by following the gradient of the values from a particular state we never reach the positive terminal state then the value of score metric is equal to the number of walkable gird cells in the map.

An interesting feature that we observed in UNet and Unet-RNN type of models is the ability to successfully use models that have been trained on smaller maps like 32x32 on much larger maps like 64x64 without need to retrain them and maintaining high success rate values. This is possible because UNet model is a fully convolutional model and after learning the value iteration algorithm can generalize it on any size of the map. Unet-RNN type of models can achieve this because they have pooling layer before RNN cell in the middle.

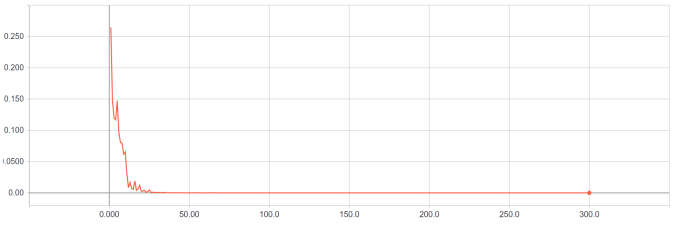

Fig. 6. Mean Squared Error test loss depending on training epoch.

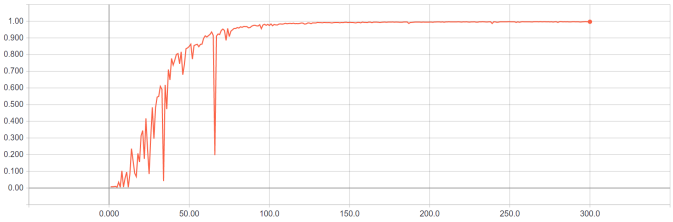

Fig. 7. Success rate metric for test data-set depending on training epoch. Value of 1.0 means that from all states in the map it is possible to reach the the positive terminal state.

D. UNet variant

The experimental results of the simple UNet variant were somewhat good with the success rate of 0.996 as seen in Table I. Looking with the eye it is almost impossible to distinguish the difference between value maps produced by neural network-based models and the Value Iteration algorithm as seen in Fig. 8. Although in order to have an optimal policy in any state of the map it would be necessary to have success rate metric of value 1.0.

In the Table II are shown a comparison between different types of non-recurrent models. Starting with Convolutional Auto-Encoder (Conv-AE), ResNet, UNet with concatenation type of skip connections and UNet v2 with addition type of skip connections.

E. UNet-RNN variant

The experimental results of the UNet-RNN variant yielded better results than UNet variant. Advantage of recurrent models is the ability to pass output map multiple times through the model to further improve precision until polices converge.

UNet-RNN, when applied to the map, are changing only a few of the closest grid cells to the wave-front of values. By changing only part of the map and not all of it at the same time, the model can predict better values than models without recurrent layers. This can be observed by watching activations of layers using Grad-CAM [14] method that allows seeing what parts of the map model gives more attention at different abstraction layers of the model.

Fig. 8. On the left ground truth generated by Value Iteration algorithm. In the middle UNet generated map. Gradients of these maps indicate state values that lead to a positive terminal state. On the right, UNet generated success map. White cells in the success map indicate that there exists a path to a positive terminal state.

TABLE I
UNET PERFORMANCE ON 32X32 SIZE MAPS.

Learning rate	Batch size	Loss	Success rate	Epoch (min)
0.001	4	3.88E-06	0.996	4.244
0.0006	4	3.38E-06	0.996	4.415
0.003	8	5.08E-06	0.996	2.719
0.002	16	6.17E-06	0.995	2.030
0.003	16	3.67E-06	0.995	2.020
0.002	8	4.12E-06	0.995	2.735
0.001	8	4.05E-06	0.995	3.076
0.001	16	4.04E-06	0.994	2.013
0.0006	8	3.25E-06	0.994	3.075
0.003	4	5.11E-06	0.994	3.698
0.002	4	4.50E-06	0.994	4.214
0.0006	16	3.97E-06	0.990	1.733

TABLE II
COMPARISON BETWEEN DIFFERENT TYPES OF MODELS FOR EMULATING VALUE ITERATION ALGORITHM.

Model	Loss	Success rate	Epoch (min)
Conv-AE	0.073	0.014	0.958
ResNet	0.002	0.054	1.195
UNet	0.001	0.956	1.807
UNet v2	0.001	0.996	2.020

F. UNet-RNN-Skip variant

Finally, we achieved close to optimal policy results with success rate of 0.998 using UNet-RNN-Skip model Fig. 9. We also compared UNet-RNN-Skip model with standard convolutional RNN auto-encoder shown in Table III as Conv-AE-RNN. It is possible to observe that skip connections have significant importance on the performance of the model. Adding additional skip connections to the architecture of the model reduced epochs needed to have the convergence of the policy.

In Table IV are shown results for UNet-RNN-Skip with different hyper-parameters. All training instances achieved success rate metric higher than 0.95. Best training instance with success rate 0.998 and loss value 3.04E-06 was with learning rate 0.001 and batch size 4. For all of the models, we used small batch sizes because it affected the stability of convergence.

TABLE III
COMPARISON OF PERFORMANCE OF UNET-RNN MODELS.

Model	Loss	Success rate	Epoch (min)
Conv-AE-RNN	8.58E-06	0.598	10.862
UNet-RNN-Skip	3.04E-06	0.998	15.833

TABLE IV
UNET-RNN-SKIP MODEL PERFORMANCE.

Learning rate	Batch size	Loss	Success rate	Epoch (min)
0.001	4	3.04E-06	0.998	19.959
0.001	8	5.20E-06	0.998	17.650
0.002	8	5.43E-06	0.997	17.459
0.003	8	1.01E-05	0.992	17.464
0.002	16	1.24E-05	0.991	15.833
0.003	4	1.92E-05	0.985	19.944
0.002	4	3.16E-05	0.983	19.753
0.001	16	1.81E-05	0.970	15.605
0.003	16	2.77E-05	0.966	15.505

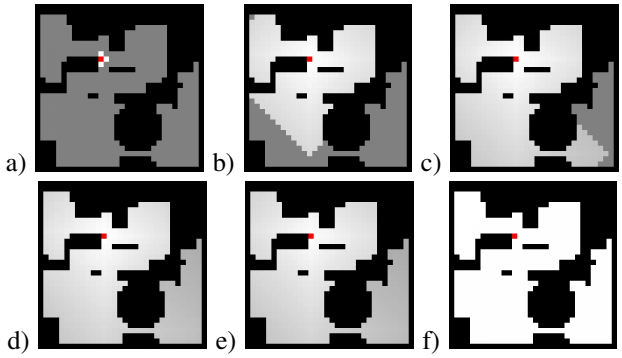

Fig. 9. UNet-RNN-Skip maps generated by progressive iterations (a, b, c), last iteration (d), ground truth (e) and success map (f).

G. Speed of convergence

Our proposed UNet-RNN-Skip model is more salable than Value Iteration Algorithm in terms of execution speed as seen in Table V. It means that the model can be used for real-time applications where recalculation of value map should be done multiple times in a second. As shown in Fig. 10 for the smallest tested map size of 32, Value Iteration Algorithm needs much more time to produce the output than UNet-RNN-Skip model. As map size increases the delta of processing time increases by average multiplier of 8 for Value Iteration Algorithm, but only by average multiplier of 3 for UNet-RNN-Skip model.

TABLE V
COMPARISON OF EXECUTION TIME IN SECONDS BETWEEN MODELS.

Model / Map Size	32	64	128	256
VI	2.95	24.873	195.902	1473.108
UNet-RNN-Skip	0.031	0.071	0.236	0.833

H. Experiments on mobile robot platform

We also tested UNet-RNN-Skip model on real-life data gathered from mobile platform using LIDAR as shown in Fig. 11 and Fig. 12. The proposed model can be used for real-time path planning and obstacle avoidance on GPU powered robotic platforms like nVidia Jetson.

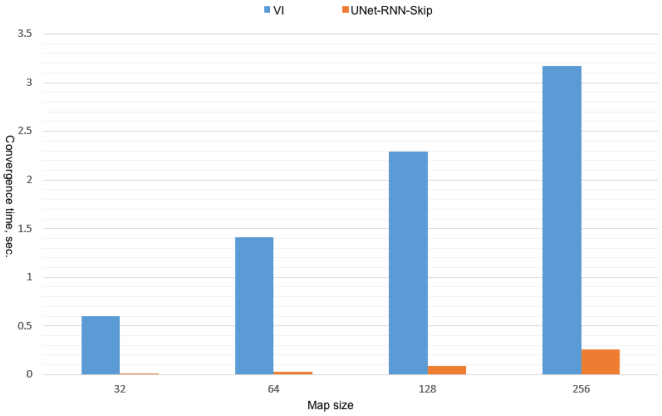

Fig. 10. Comparison of time to convergence between models on different size maps (log scale).

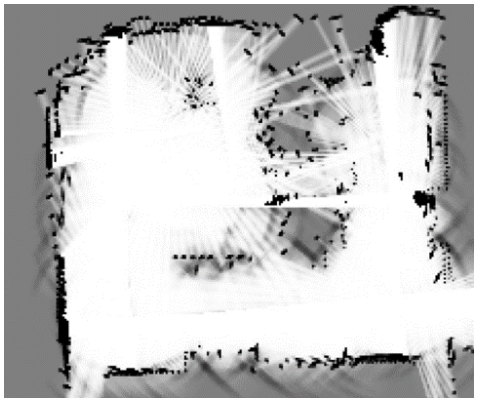

Fig. 11. Map generated from LIDAR point cloud on 2D plane from mobile robot platform.

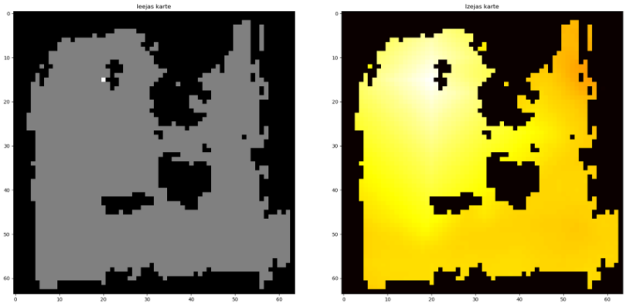

Fig. 12. Discretized map from LIDAR point cloud on 2D plane on the left. The map of values of states after passing through UNet-RNN-Skip model on the right.

V. FURTHER RESEARCH

The proposed model could be used also in other applications where input is sequential data and the task is to transform part

of it or to segment it depending on previous time steps. It can be used to model all sorts of sequential algorithms with input that could be represented as a matrix with spatial features. Some example use-cases could be colouring task of movies or tracking and segmentation of moving objects in between frames. Also, these models have been successfully used in spectral processing tasks within the realm of audio processing in commercial product http://www.asya.ai.

Our synthetic data-set generator could be used in other path planning or mapping problems research.

Another line of research could be the development of models that work on different size maps. Currently, our models have been tested on 64x64 maps when they have been trained on 32x32 maps, but it would be interesting to research the limits of such model ability to generalize on even larger maps.

VI. Conclusions

This research shows that UNet-RNN-Skip models can be used to parallelize Value Iteration algorithm and achieve comparable results in shorter execution time. New synthetic data-set has been introduced and source code to generate even more data-sets that could be used in further research. Results show that on GPU powered robotic platforms UNet-RNN-Skip models could be used in real-time whereas sequential Value Iteration algorithm would be impractical.

Acknowledgments

Research has been completed with a support from High-Performance Computing Center of Riga Technical University.

References

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, Dec. 2002.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2015.

[4] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," in *AAAI*, 2016.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[6] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *MICCAI*, 2015.

[8] A. Tamar, S. Levine, P. Abbeel, Y. Wu, and G. Thomas, "Value Iteration Networks," in *IJCAI*, 2016.

[9] C. Kamanchi, R. B. Diddigi, and S. Bhatnagar, "Second Order Value Iteration in Reinforcement Learning," *arXiv:1905.03927 [cs, stat]*, May 2019.

[10] D. Schleich, T. Klamt, and S. Behnke, "Value Iteration Networks on Multiple Levels of Abstraction," *arXiv:1905.11068 [cs]*, May 2019.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

[12] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[13] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," p. 4.

[14] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017.

APPENDIX B - Paper 2

Chapter 3

ChromSword®: Software for Method Development in Liquid Chromatography

Sergey V. Galushko*‡, Irine P. Shyskina*, Evalds Urtans† and Oksana Rotkaja†

*ChromSword, Dr. Galushko Software Entwicklung GmbH Im Wiesengrund 49B, 64367, Muehltal, Germany

†ChromSword Baltic, Antonijas 22-1, Riga, LV-5041, Latvia

‡galushko@chromsword.de

3.1 Introduction

Method development in chromatography can be considered as a process studying the empirical relationships between the quality of a chromatogram and the chromatographic conditions. A chromatographer changes conditions to find an acceptable method to achieve separation in a reasonable time. The time required to find optimal conditions or to make any conclusion can be substantially reduced by using computer programs for method development. HPLC method development programs can be utilized interactively (off-line) and for automatic optimization (online). ChromSword® for off-line computer-assisted method development was launched in 1994 as an extension of ChromDream® software [1]. During 1998–2000, the first version for unattended method development was started [2]. The latest version of ChromSword® combines different technologies of method development in one software platform:

- Computer-assisted
- Automated optimization

- Automated robustness studies
- Scouting to screen different column, solvents, buffers and methods

It is possible for a chromatographer to use only the computer-assisted (off-line) or automated method development approach or to use both interactive and unattended optimization.

ChromSword® off-line can be used for optimizing separations in reversed-phase (RPLC), normal-phase (NPLC) and ion-exchange (IEX) liquid chromatography (LC). In the off-line mode, chromatogram simulations and optimizations as a function of one or more variables are possible. The off-line mode includes two possibilities for optimization in RPLC.

The approach which takes into account the characteristics of compounds and column/solvent properties is the solvatic or solvophobic model of RPLC.

The traditional method for optimizing separation using only retention data of analytes is the linear solvent strength (LSS) model and other polynomial models.

In the automated mode, the software operates as a chromatography data system controlling HPLC instruments and executes a sequence of runs. The user can predefine such a sequence of runs — this is a scouting approach to screen different stationary phases (SPs) or mobile phases (MPs) or statistical design of experiments (DoE) according to some statistical rules to study the effect of method variables on the separation. This method is defined as robotic process automation. Another approach is intelligent automation. Intelligent automation automates non-routine tasks like optimizations involving complex data processing and reasoning. ChromSword® supports both types of automation to assist chromatographers for routine and intelligent method development workflow.

To support various method development workflows ChromSwordAuto® package contains modules dedicated to different scenarios and tasks:

ChromSword®	for computer-assisted method development
ChromDraw®	chemical editor for drawing and processing structural formulae

ColumnViewer®	reversed-phase column properties data base
ChromSword® Scout	for automated method screening
ChromSword® Developer	for automated method optimization
AutoRobust®	for automated robustness study and method transfer
ReportViewer®	for data browsing, chromatogram and spectra processing, project management and report generation

3.2 Automated Method Development

Most automated HPLC method development approaches can be divided into three classes:

- Mechanistic or model-based optimization.
- Statistic or direct process optimization.
- Screening or running a large number of column/solvent/method combinations to identify those with a reasonable separation.

In the model-based optimization, mathematical models are utilized to reduce the number of experiments. The development of mechanistic models requires good chromatography understanding, reliable tests for parameter estimations and peak tracking. Limiting factors are computational time and reliability of the models that are applied for simulation and optimum search. The determination of mechanistic model parameters can be complicated for computer-assisted (off-line) method development and requires time and operator qualification for optimization of multi-component mixtures. Automatic optimization with mechanistic DoE incorporates engineering knowledge in the form of constrains, expert-rules and known fundamental relationships of LC; therefore, this technology can find optimal conditions faster than the off-line approach. One of the main advantages of the automatic optimization is that a chromatographer can avoid complex tasks of the off-line computer-assisted optimization — peak tracking, data input, method and sequence specifications and other routine and non-routine operations. It should be noted that in the

recent final guidance for industries with regard to the analytical method development, the U.S. Food and Drug Administration (FDA) recommends submission of data to indicate a mechanistic understanding of the basic methodology [3].

An alternative to the mechanistic model-based approach is to directly identify process optima based on the results of experiments that are planned by statistical software such as repeated DoE. In contrast to model-based strategies, no mathematical process model is required, which is a significant advantage for many operators, and it is also better to use when the theory of LC and separation process interactions are not yet fully understood. Unfortunately for complex mixtures, when retention models cross each other in different regions of method variables, the direct approach can find the optimum only accidently. Usually, this type of DoE is used in a case where no, or little, prior process knowledge is available. However, for separation processes where a high degree of knowledge is available, statistical DoE is often not the most efficient strategy. Nevertheless, experimental results from the direct approach can be successfully used to identify a local optimal separation region for simple mixtures and to estimate the sensitivity of method quality to specific parameter changes within the design space (DS). Special software that include both features to create DoE and control of LC instruments to execute the DoE have substantial advantages against statistical software which have only options to plan DoE.

An alternative to the mechanistic and statistic approaches is to run the high-throughput screening to test combinations of method variables and factors — columns, solvents, buffers, gradients, etc. In contrast to the model-based and the statistical strategies, neither mathematical process model nor statistical DoE is required for the scouting approach. A chromatographer needs to only create a large sequence and then run it for new samples, thus relying on these few combinations of method variables and factors that will provide practically reasonable separations. The scouting approach is used frequently for chiral separations and samples when specific optimization is not necessary. Specialized software for automated method scouting are practically useful to create and edit long sequences rapidly and run them automatically.

For analytical method development, all three approaches proved to be practically useful, and any combination of them increase the probability of finding more suitable methods. To support various automated method development workflows, ChromSwordAuto® can operate in three modes: scouting, model-oriented optimization and statistic (direct optimization). Each mode can be applied separately or in various combinations depending on the preferred strategy of method development at a particular laboratory and project stage. Each mode is operated with a dedicated module.

3.2.1 *Instrument control and software configurations*

ChromSwordAuto® can operate as a chromatography method development data system (CDS) or as a third-party software. Functioning as the CDS ChromSwordAuto® controls Agilent, Waters and Hitachi HPLC and UHPLC systems. To control these instruments, no other CDS is necessary, and a stand-alone or a client–server configuration of ChromSwordAuto® can be chosen during installation. For the client–server configuration, data are collected on the local network or the internet file server (Fig. 3.1). The client–server configuration satisfies the requirements for data integrity with regard to applicable regulations like FDA 21 CFR Part 11.

Operating as a third-party software, ChromSwordAuto® controls Agilent, Waters and Dionex instruments thorough OpenLab/ChemStation, Empower or Chromeleon CDS. These CDS can work in the stand-alone, network or client–server environments.

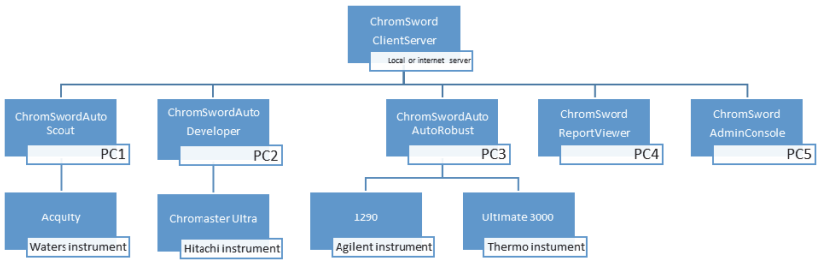

Figure 3.1: ChromSwordAuto® client–server configuration.

Different configurations of HPLC and UHPLC instruments can be used for automated method development. The most simple method development system consists of a binary pump, UV detector and autosampler; however, typically, method development systems contain 4–8 columns and 2–6 solvent channels to test different stationary and MPs.

ChromSwordAuto® incorporates automation of routine operations: column equilibration, column wash-out methods, system purging and column and solvent switching sequences.

3.2.2 *Strategies of automated method development*

Different strategies can be applied for automated method development. Strategies can combine screening, optimization and robustness study steps. One of the successful strategies for development of RPLC methods with ChromSwordAuto® has been used for dug candidates. It includes an automated screening step to identify the best column and solvent followed by an optimization step to fine-tune the separation [4, 5]. A similar strategy was used to apply ChromSwordAuto® for optimization of chiral separations in NPLC [6] and RPLC [7]. In another approach, the rapid optimization mode can be used for several predefined SP and MP combinations which are accepted at a lab as a standard method development column set, and then the fine optimization mode is applied for the most promising combination. Robustness studies can be included optionally for late-stages projects or methods to be transferred to other laboratories. The steps of such a strategy are shown in Fig. 3.2.

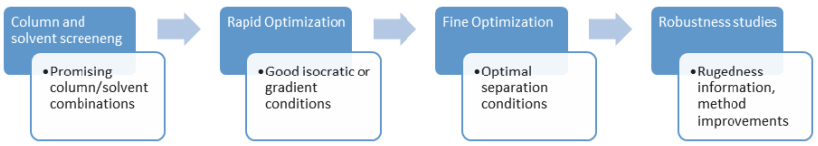

Figure 3.2: The strategy of method development for the latest stages of product developments.

3.2.3 *Automated method screening with ChromSwordAuto® Scout*

Automated screening of SP and MP are used to find practically a acceptable separation and run time when full optimization is not necessary. The screening can also be the first step in a multi-step method development strategy to identify promising combinations of columns and MPs.

ChromSwordAuto® Scout screening module generates sequences automatically and runs them to scout different gradients, columns, solvents, buffers, temperatures and other method variables for one or several samples. For multi-column and multi-solvent instruments, ChromSwordAuto® Scout controls several column compartments with 4–8 columns in each compartment and several (4–12 position) solvent switching valves connected to a binary or a quaternary pump. ChromSwordAuto® Scout analyzes 2D and 3D data acquired from two detectors simultaneously.

ChromSwordAuto® Scout application incorporates automation of column equilibration, column wash-out methods, system purging and column and solvent switching sequences for changing solvents, buffers, columns and other chromatographic process variables and factors.

3.2.4 *Automated model-based method optimization with ChromSwordAuto® Developer*

ChromSwordAuto® Developer module can be used for automated method optimization in RPLC, NPLC, IEX, HIC, HILIC, size exclusion chromatography (SEC) and supercritical fluid chromatography (SFC). For SEC, ChromSwordAuto® optimizes isocratic conditions, and for another type of chromatography, both isocratic and gradient separations can be optimized. Retention models that are used for different type of LC are described in Section 3.3.

ChromSword® is used for automated optimization of various mixtures; however, most frequently, it is applied for method development in the pharmaceutical industry. Typical applications are the development of stability-indicating and quality control methods (e.g. impurity profiling,

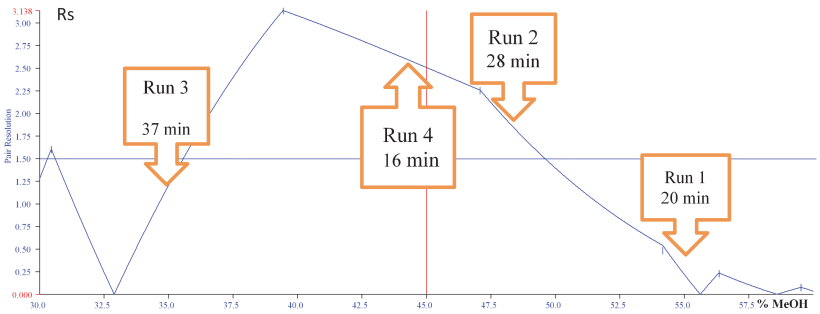

Figure 3.3: Runs shown on the resolution map that the software performs searching for optimal conditions in the unattended mode. Method development for a mixture of nine beta-blockers. Column: Purospher RP 18e, 5 μm, 150×4 mm. Mobile phase: 0.05 M phosphate buffer, $\mathrm{pH} = 3.0$ — methanol. The goals: $R_s \geq 2.0$ and run time ≤ 20 min.

assay, cleaning control, etc.). For automatic optimization, a user should specify the starting conditions: the column, solvent, flow rate, injection volume and the task type — rapid or the fine optimization. A chromatographer can also specify the development of either isocratic or isocratic and gradient methods. For both procedures, the optimization process includes the study of a sample to build retention models followed by application of the optimization procedure to find the optimal conditions. For planning new runs, the software processes the results of the previous runs and takes them into account. In Fig. 3.3, the method by which the software searches for optimal conditions developing the isocratic methods is shown.

For optimizations of gradient methods, both the studying and optimization runs can be linear and multi-step gradients. For optimization of separation, the Monte Carlo, genetic algorithms and the neural network methods are used. For the rapid optimization algorithm, the software performs 3–4 runs (Figs. 3.4–3.6), and for the fine optimization algorithm more runs are executed to study a sample and optimize the separation.

3.2.4.1 *Method development for large molecules*

Large molecules like proteins exhibit substantially different retention behavior than small analytes [8]. For these samples a small shift in chromatographic conditions can lead to high changes in retention and

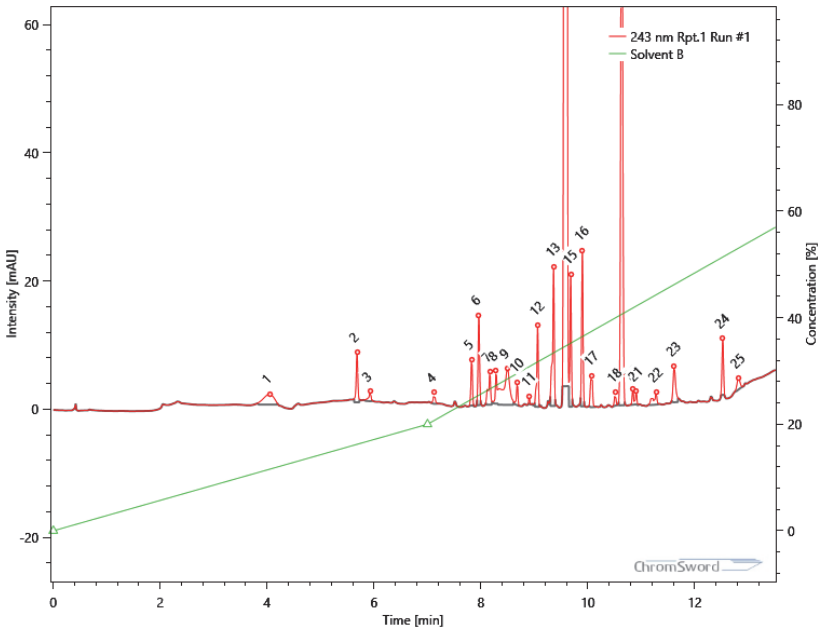

Figure 3.4: The first run of the automatic rapid optimization of the force degradation test mixture. Column: Zorbax Eclipse C18, 1.8 μm, 50×2.1 mm, flow rate 0.6 mL/min.

efficiency. The other point is that these compounds have practically identical UV spectra and cannot be used for peak tracking. Recently computer-assisted (off-line) method optimizations were reported for monoclonal antibodies (mAbs) and their domains in RPLC and IEX using 2D model as the gradient time–temperature model [9,10]. It should be noted however, that the computer-assisted method optimization can be a time consuming process when many samples, columns and effects of different method variables require evaluation. An effective approach to circumvent and increase productivity is automated method development. In this instance, an analyst defines a strategy and an '*intelligent*' chromatography method development data system plans and performs many routine and optimization experiments autonomously. Various strategies of automated method development for mixtures of large molecules can be realized with ChromSwordAuto®. These can combine automated screening experiments with unattended

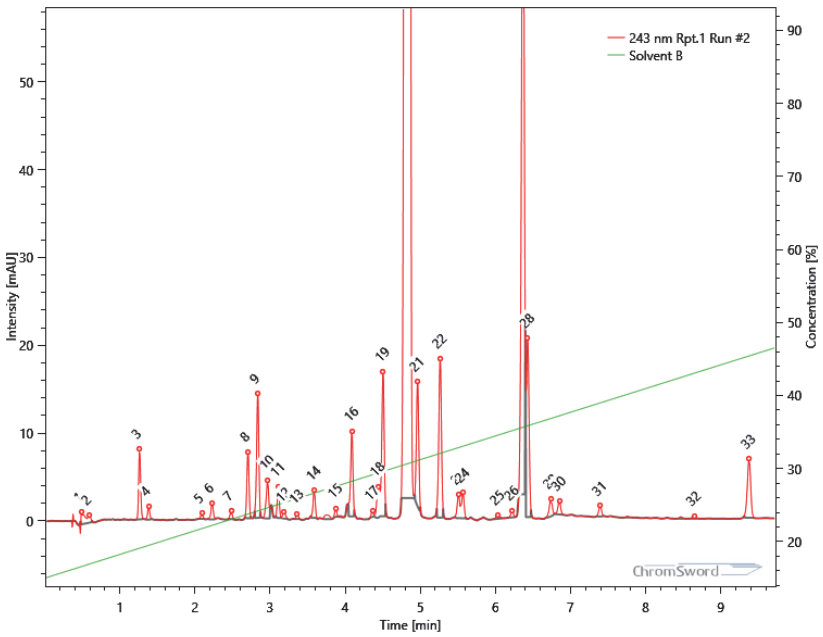

Figure 3.5: The second run of the automatic rapid optimization. Conditions are same as described for Fig. 3.4.

optimization, which is then followed by robustness studies using different DoEs. Results can also be used for off-line simulation and optimization. Such a strategy is used in different laboratories for automated RPLC method development using ChromSwordAuto® for the separation of variants and degradation products of the recombinant mAbs. The aim of method development for such projects is to study the domain-specific oxidation and develop stability-indicating methods that separate degradation products. For complex mixtures the optimization program can run multi-step gradients to separate more components (Fig. 3.7).

An important point to be considered is the column length for optimization of small and large molecules. It is known that the column efficiency for small compounds like peptides, after the digestion of proteins, is improved by increasing the column length. In contrast, the retention behavior of large proteins is different, and their bandwidth can be almost

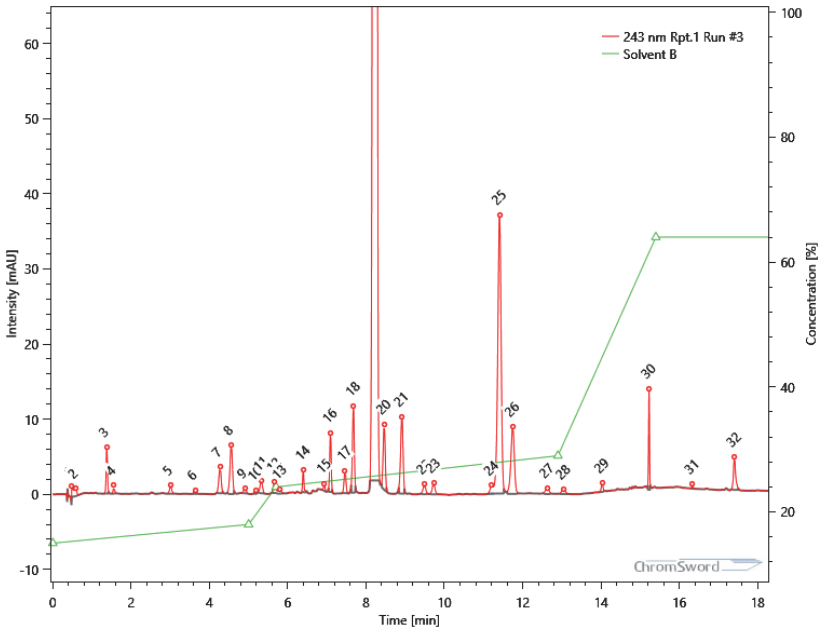

Figure 3.6: The third run of the automatic rapid optimization. Conditions are same as described for Fig. 3.4.

constant for all practical column lengths in the range 50–250 mm [11]. For such samples, longer columns do not provide higher separation efficiency [11], and therefore a short column can be a good alternative. Results in Figs. 3.7 and 3.8 show that the automated procedure can successfully find conditions to separate proteins on small columns. It should be noted that the optimization procedure is not related strictly to the column length. It is related to the target resolution and practical run time; therefore, shorter run times can be obtained on a long column and longer run time on a short column. In Fig. 3.8(a) the initial three study runs and in Fig. 3.8(b) the final gradient run are shown to separate monoclonal antibodies, under RPLC conditions. It should be noted that no optimal linear gradient for this mixture could be found in the temperature range of 70–80°C where reasonable peak width is observed and the column can be operated.

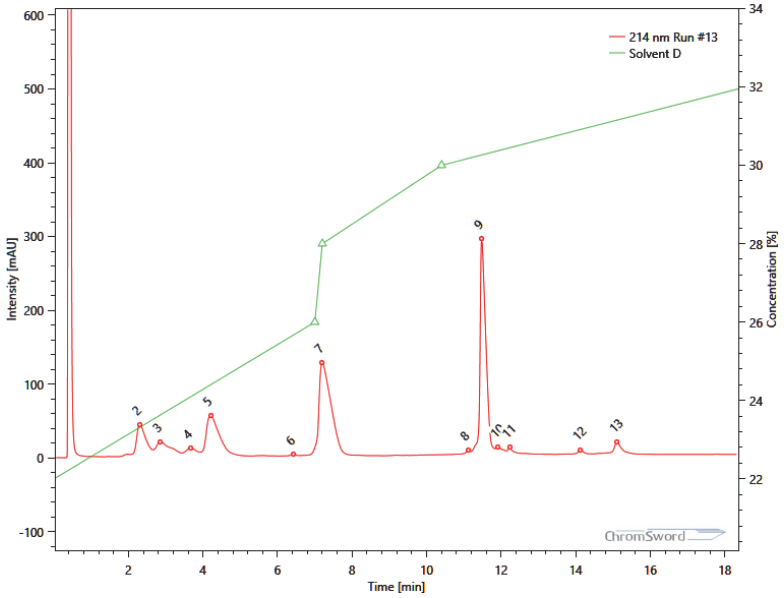

Figure 3.7: Partially digested (using IdeS) and reduced (using dithiotreitol, DTT) mAb sample. Peaks 2–4 — oxidation products of the crystallizable fragment (Fc/2); peak 5 — (Fc/2); peak 7 — the light chain (LC); peak 9 — the N-terminal half of one heavy chain (Fd). Column: 50 mm × 2.1 mm AdvanceBio RP mAb C8. Mobile phase A: Water + 0.1% TFA, B: ACN + 0.1% TFA. Temperature was set to 70°C, flow rate = 0.3 mL/min.

3.2.5 *Automated robustness studies and statistical DoE with ChromSword® AutoRobust*

ChromSword® AutoRobust is a specialized application for automatic evaluation of robustness of HPLC methods. According to the ICH guidelines [12] "Validation of Analytical Procedures: Methodology (Q2B)," the robustness of an analytical procedure is defined as a measure of its capacity to remain unaffected by small, but deliberate variations in method parameters and provides an indication of its reliability during normal usage. The robustness should be considered at an appropriate stage in the development of the analytical procedure [12]. AutoRobust is a software tool for automation of robustness experiments to study the influence of variations in method parameters on chromatographic results.

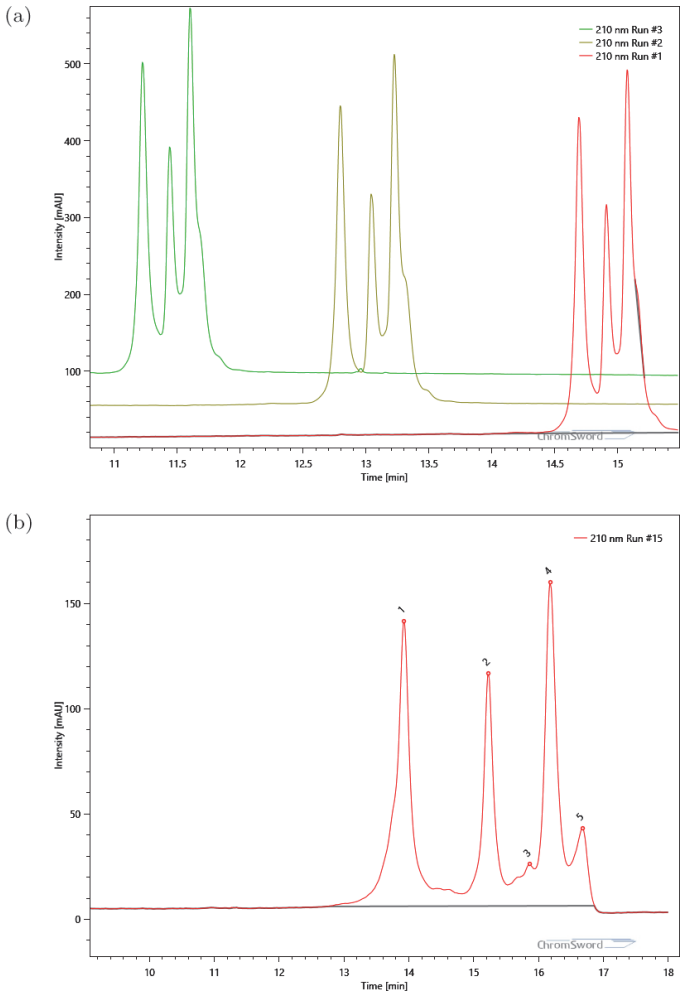

Figure 3.8: Column: 50×2.1 mm Zorbax 300 SB-Diphenyl. Mobile phase A: water + 0.1% TFA, B: ACN + 0.1% TFA. Flow rate: 0.25 mL/min; Temperature: 80°C. Sample: test mixture of mAbs (mAb1, mAb2 (confidential), Erbitux and Avastin). (a) Initial study runs of unattended optimization for separation; gradients: 1. 30–70% B in 25 min; 2. 36–66% in 22 min; 3. 36–66% in 19 min. (b) The final run of the unattended optimization; gradient: 0 min — 50% B in 2.2 min — 51% B; 16.6 min — 54% B; 18 min — 55% B.

Robustness of a method is extremely important for providing method transfer to other laboratories and instruments. Typically, robustness tests are performed at late stages of drug development projects; however, performing robustness tests at later stages involves the risk that when a method is found to not be robust, it should be redeveloped and optimized. Therefore, it is better to perform robustness tests at an earlier stage of method development. Different critical quality attributes (CQAs) of a method can be tested — including area, area%, retention time, resolution and other CQAs. One of the most important CQAs for HPLC methods is the resolution between peaks of target compounds. The resolution characteristic of a method should be within appropriate limits to ensure the drug product quality.

The following steps can be identified for robustness tests projects:

(1) selection of the factors to be tested,
(2) selection of the experimental design,
(3) definition of the different levels of the factors,
(4) creation of the experimental set-up,
(5) execution of the experiments,
(6) calculation of effects,
(7) statistical and graphical analysis of the effects,
(8) drawing conclusions from the statistical analysis and
(9) if necessary, improving the performance of the method.

These different steps are considered in more detail below.

3.2.5.1 *Selection of the factors*

For robustness tests, different operation factors can be considered. The selected factors can be quantitative (continuous) like the temperature or the concentration or qualitative (discrete) like the column batch. These factors should represent those that can be changed when a method is transferred between laboratories, analysts or instruments and that potentially could affect the response of the method. Typically, the following factors can be included in the robustness tests:

- gradient time and slope of linear gradients,
- initial and final concertation of linear gradients,

- time and concentration of each gradient node (step) for multi-step gradients,
- flow rate,
- column compartment temperature,
- pH of the MP,
- wavelength,
- column batch,
- method equilibration time,
- injection volume.

All these parameters and factors are supported by automated DoE with ChromSword® AutoRobust module. A chromatographer can optionally specify all or several factors to be included in the DoE.

The difference in flow rate, concentration and gradient time affect the resolution when different type of pumps (low- or high-pressure mixing systems), different solvent mixers and pumps from different manufacturers are used. The effective temperature inside a column can be different due to the difference in construction of compartments (forced air or still air oven). The small difference in glass electrodes and standard buffers can lead to differences in pH of a MP and selectivity of separation of basic and acidic compounds. If concentration of a sample is too low or too high, then increasing the injection volumes can lead to peak distortion.

3.2.5.2 *Selection of the experimental design*

The one-factor-at-a-time (OFAT), full factorial design (FFD) and the Plackett–Burman partial factorial design (PBD) can be used for robustness tests. The OFAT is the fastest design; however, it cannot estimate interactions of different variables without preliminary studies. The FFD is the most comprehensive design to determine interactions of factors and describe the response surface for finding optimum factor-values; however, it requires substantially more experiments. The PBD can be used as an alternative to FFD, but arrays of data points after the PBD cannot typically be used to solve the system of equations to determine chromatographic retention model parameters. In this case, a less reliable, simplified model is usually used to calculate response; however, deviations between the predicted and experimental value of a critical quality parameter can be too high. Another

problem is a possible confounding of effects due to reducing the number of runs in PBD. In this case, the effects of different factors or interaction factors cannot be evaluated individually and the interpretation of the results becomes difficult and even incorrect.

We consider that the robustness projects should include two designs:

(1) The OFAT design which can rapidly identify which of tested variables has a significant effect on the response.
(2) The FFD of the critical variables which were identified in (1).

Both steps can be executed in a completely automatic manner with a reasonable number of experiments. The PBD can be planned when the number of runs is too high and it is not practically reasonable to run the FFD designs.

3.2.5.3 *Definition of the levels for the factors*

The factor levels of variables to be tested should be set around the nominal values specified in the operating (basic) method. The interval chosen between the extreme values represents the limits between which the factors are expected to vary when a method is transferred. It should be noted that the levels should be defined by the analyst according to the results of a preliminary study of chromatographic retention behavior of compounds and instrument specifications taking into account the precision and the uncertainty with which a factor can be set and reset. To define the factor levels for the temperature, concentration and time of gradient steps, it is recommended to study the effect of these variables in more detail.

3.2.5.4 *Creation of the experimental set-up*

Each variable is studied in the experimental design, which is selected as a function of the number of factors and of levels to investigate. Two-level screening designs are a simple approach that can screen a relatively large number of factors in a relatively small number of experiments. More informative are the two-level designs with center points for effects of concentration and gradient time or the four-level designs with center

points for effects of flow rate and temperature. Such designs are optional in AutoRobust and allow the analyst to establish a linear or nonlinear retention model. Creation of the experimental design manually takes substantial time, even for OFAT. For planning FFD and PBD, normally special statistical software are used and then the design plan should be transferred into a sequence of runs of a chromatography data system. This is also a time-consuming process, and is practically very important that robustness test software can create DoE and transfer it into a sequence of runs automatically. The AutoRobust software module in ChromSword® provides a simple and rapid automated set-up of up to eight variables with 2–7 levels for OFAT, FFD and PBD. An unlimited number of qualitative factors (column, solvent batches, etc.) can also be included in the DoE.

3.2.5.5 *Execution of experiments*

It is important for reproducible robustness experiments to provide constant parameters both for injection and conditioning runs. Column and instrument wash-out, and purging and conditioning runs should be set up according to the instrument and column specifications. Adequate time for column equilibration, not less than 10 column volume have a paramount importance especially for large proteins to obtain reproducible results. For more confidence, it is recommended to include the column equilibration time as a variable in the robustness tests DoE.

The planned DoE is executed automatically with AutoRobust. The method development system performs these runs while interacting with a chromatography data system or directly with the modules. For estimation of time effects and stability of the instrument and the column, a number of additional experiments at nominal levels can be added to the planned DoE. These replicate experiments are performed before, at regular time intervals between, and after the robustness test experiments. These experiments allow checking whether the method performs well at the beginning and at the end of the experiments and to estimate for drift and column stability.

The results of runs are used to calculate effects of variables and determine the response.

3.2.5.6 *Calculation of effects and response determined*

From the performed experiments, a number of responses can be determined. For chromatographic methods, responses describing a quantity such as the content of main substance and by-products and effects of variables on peak area% and areas should be evaluated. The responses determined during the robustness test can be one of the following: the resolution between each pair of neighboring peaks, the retention time, the area and the area% of compound peaks. These parameters allow for evaluating the quality of a method and the effects of variables and factors.

The automated data processing procedure additionally calculates the relative retention, the peak asymmetry, the peak height and number of theoretical plates, which can also be included in the robustness study results.

3.2.5.7 *Numerical and graphical analysis of the effects*

One of the most important CQAs for HPLC methods is the resolution between peaks of target compounds. The resolution characteristic of a method should be within appropriate limits to ensure the drug product quality. As mentioned earlier, two approaches can be used to evaluate the effect of method variables on resolution — descriptive and mechanistic. Traditional statistically based software uses the descriptive approach and models the response surfaces with quadratic polynomials [12]. The main advantage of this approach is the simple and easy data processing procedure. This approach does not use physical models of the separation process and peak tracking from run to run. However, from the theory and practice of computer-assisted HPLC method development, it is well known that the quadratic dependence between resolution and method variables (concentration of organic modifier, gradient profile, temperature, pH) is more an exception rather than a rule for complex mixtures with irregular retention models [8]. Retention models of compounds can cross each other, and dependences $R_s = f$ (temperature, concentration, gradient time, pH) can have one or several maxima and minima. Figure 3.9 shows the resolution plots for limited pairs of a mixture of nine beta-blockers as a function of the concentration of methanol in the mobile phase. It is obvious that

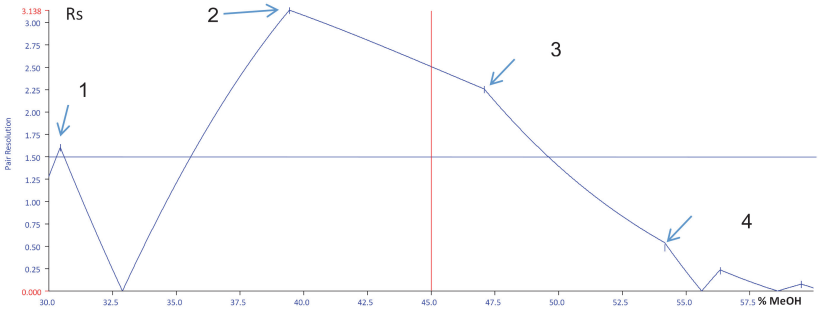

Figure 3.9: Resolution map: Effect of methanol concentration in MP on resolution of a mixture of nine beta-blockers. The arrows show the change of the limited pair in different regions of methanol concentration. 1 metipranolol/alprenolol; 1–2 propranolol/metipranolol.; 2–3 carazolol/celiprolol; 3–4 metoprolol/celiprolol; alprenolol — carvedilol.

modeling of the resolution response without peak tracking in this case will lead to wrong conclusions regarding optimal conditions and robustness of the method. The mechanistic approach uses parameters of the chromatographic process responsible for the response; however, retention behavior of the compounds must be studied to describe the effect of variables on the resolution. These include peak tracking from run to run, evaluation of parameters of retention modes in gradient elution and under different temperatures, and building a system of equations and solving them.

The mechanistic approach that applies relations from the theory of LC is supported in the AutoRobust software. After the design of experiments is created and performed in automated mode, data are processed for statistical and graphical analysis of responses. Method variables can have a substantial effect on resolution, and knowledge of the effect of the combination of these variables is necessary to study the robustness and to build up a DS of the method. The example of the effect of two variables with a fixed nominal value for two other variables is shown in Fig. 3.10.

3.2.5.8 *Improving the performance of the method*

Analysis of the resolution maps for a combination of three different variables enables visualization of areas where resolution can be increased or decreased. For example, the resolution map shows that temperature

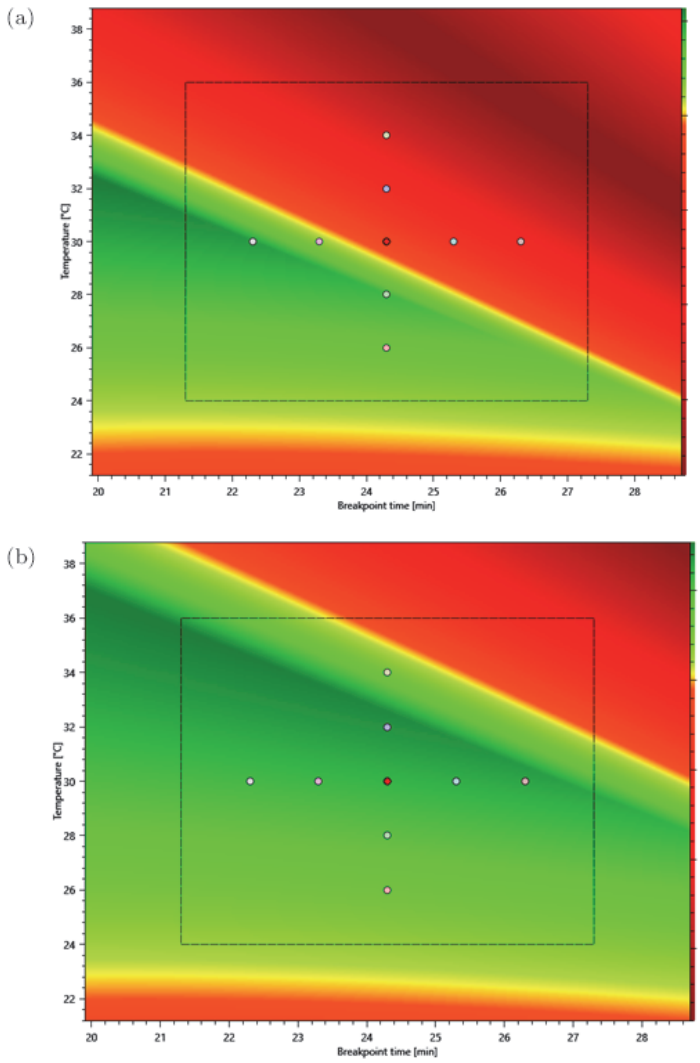

Figure 3.10: Resolution maps: Effect of the temperature and the gradient breakpoint time on resolution of a limited pair at the flow rate of 1.0 mL/min (a) and 0.8 mL/min (b). Mixture: 10 hair dyes. Column: ACE Excel C18-Amide 100×4.6 mm, 3 μm.

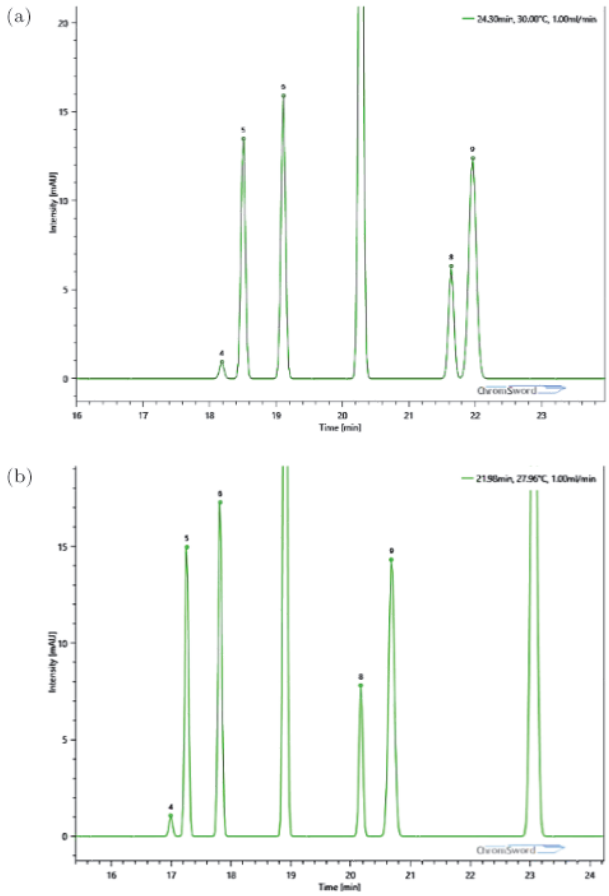

Figure 3.11: Chomatograms at a temperature 30°C, flow rate of 1.0 mL/min and gradient time of 24 min (a) and at 28°C, 0.80 mL/min and 22 min, respectively (b).

at 28°C, the flow rate of 0.80 mL/min and the gradient time of 22 min will provide a more robust method with higher resolution than one that was used after optimization (30°C, 1.0 mL/min and 24 min, respectively) (Figs. 3.10(b) and 3.11). Thus, robustness studies can also be considered as an additional tool to improve the performance of the method.

3.3 Computer-assisted Method Development

ChromSword® in the off-line mode can be used for optimizing separations in RPLC, NPLC and IEX.

If the structural formulae of compounds are known then, ChromSword® can predict the conditions of isocratic or gradient elution for acceptable retention to be obtained. No preliminary experiments need to be performed for the virtual chromatography. If the structural formulae of compounds being separated are known, then it is possible to start optimization of resolution after the first run. In this case, after inputting the experimental retention data for the first run, parameters of solutes will be refined to predict the best conditions for the separation. Entering experimental retention data for the second and the following runs makes possible a more precise prediction.

For solutes with unknown structures, ChromSword® can determine, from chromatographic experiments, their characteristics (molecular volume, the energy of interaction with water, nature (acid, base, neutral, p*Ka* value) and then predict their retention times on different reversed-phase columns and with different MPs.

Prediction is the first step in method development. The subsequent steps are optimization of retention and separation. ChromSword® enables a user to optimize the concentration of a modifier in a MP, pH value, temperature, gradient profile and column coupling. To optimize the separation of a mixture in gradient elution mode, stochastic methods like Monte Carlo and genetic algorithms are used.

For NPLC, it is possible to optimize the concentration of a stronger solvent in a weaker one when the retention data for two or more runs are entered. For IEX, the buffer or salt concentration in a MP can be optimized. Optimization of temperature is possible both for NPLC and IEX.

Optimization of method variables are organized in different modules of the software. The results depend on the information that a user enters into the software (Table 3.1).

ChromSword® can work with massive amounts of data. One sample file can contain up to 100 compounds including structural formulae and the data for up to 20 runs in the each module.

Table 3.1: Input/output of ChromSword® in the off-line mode.

Minimal input	Expected output
Structural formulae are considered	
Structural formulae (up to 100 in a file)	Starting conditions for RPLC: column type, eluent.
Structural formulae and data of one run	Optimal eluent for separation of a mixture in isocratic RPLC on a column being used. Optimal gradient profile. Starting conditions of RPLC for other column types and an eluent.
Structural formulae are not considered	
Data of two runs with different concentrations of an organic solvent in a MP (RPLC)	Optimal eluent for separation of a mixture in isocratic RPLC on a column being used. Starting conditions of RPLC for other column types and an eluent. Evaluation of the analyte parameters (molecular volume, polarity).
Data of two runs with different concentrations of an organic solvent or a buffer in a MP	Optimal eluent for separation of a mixture in isocratic RPLC, NPLC and IEX. Optimal gradient profile.
Data of two runs with different gradient profiles	Optimal gradient profile for separation of a mixture in gradient HPLC. Optimal eluent for separation of a mixture in isocratic HPLC.
Data of two runs with different temperatures of a column	Optimal temperature for separation of a mixture in isocratic HPLC. Enthalpy sorption of analytes.
Data of two runs with different pH of a MP	Optimal pH for separation of a mixture in isocratic RPLC.
Data of three runs with different pH of a MP	Optimal pH for separation of a mixture in isocratic RPLC. Nature of analytes (base, acid, neutral). pK value of analytes.
Two variable optimizations	
Data of three and four runs with different concentrations, pH, temperatures, columns, solvents, gradient profiles	Optimal gradient profile and temperature; concentration and pH; concentration and temperature; pH and temperature; concentration of two different organic solvents; optimal connection of two columns with different selectivity and concentration, gradient profile, pH or temperature.

3.3.1 *Concepts and procedures for developing HPLC methods*

The central idea of the computer-assisted method development is to input information about the mixture to be separated and then to apply a computer simulation to predict results for different chromatographic conditions, thus finding the acceptable conditions for separating the mixture.

One of the options is to use structural formulae as input for a computer program and to predict acceptable chromatographic conditions by analyzing information concerning their structures. It is an easy way for the user, but it is one of the most complicated problems in chromatographic science to predict acceptable conditions from a chemical structure. A much less complicated problem is to predict the results of chromatographic experiments by analyzing the results of several experiments previously performed.

It is understandable that the less information a computer program receives, the less precise the prediction that is obtained. If the input is only the structural formulae of compounds, the level of predictability is much less than that we would have after entering the results of several chromatographic experiments and their conditions. On the other hand, the fewer experimental results the computer program requires to produce acceptable prediction, the less time we have to spend developing the method.

It is hard to obtain an exact prediction of the retention time values from the structural formulae. The task of working with structural formulae is not to enable the precise prediction of retention in the first-guess experiment but to predict the concentration of an organic solvent in a MP (or a gradient profile) for acceptable retention to be obtained. Successful prediction of the concentration or the gradient profile will save time and the amount of solvent used in the experimental work. From a practical point of view, it is not important at this stage to predict the retention factor values precisely. The most important issue is to obtain these values within the acceptable practical limits of 1–20.

A practically reasonable approach is to start method development with only the information about structure, to receive the first prediction of chromatographic conditions (the first-guess method), to inject the sample and then to use experimental retention results for correcting the first-guess

prediction. In this case, a good chance exists to find acceptable conditions within a minimal amount of time. However, in many cases, a chromatographer has no information about compounds in a mixture or the structure parameters are not known. This situation is typical for developing stability indicating methods, reaction monitoring, separation of bio-mixtures and large molecules. In this case, it is necessary to obtain retention times for two or more experiments and then start computer experiments.

3.3.2 *Retention models*

The retention model in ChromSword® is defined as a type of a mathematical equation which describes the relationship between the retention of a compound and its properties as well as the conditions appertaining to the chromatographic experiments.

It is the focal point in method development software to determine retention models that adequately describe the effect of chromatographic conditions on the retention of compounds in a sample. In this case, based on only a few experiments, the software can predict the results of many other experiments under different conditions, thus allowing a chromatographer to simulate experiments with a computer and find the conditions for acceptable or best separation.

ChromSword® supports two approaches for the determination of retention models in RPLC. These are as follows:

(A) A traditional formal approach which applies linear, quadratic, cubic or other polynomial models for describing the relationship between the retention of solutes and the concentration of an organic solvent in a MP:

$$\ln k = a + b(C) \tag{1}$$

$$\ln k = a + b(C) + d(C)^2 \tag{2}$$

$$\ln k = a + b(C) + d(C)^2 + e(C)^3 \tag{3}$$

where k is the retention factor of a compound, C is the concentration of an organic solvent in a MP and a, b, d, and e are parameters of equations that must be determined by the software for each compound from the retention data obtained by using different concentrations of an organic solvent in a MP.

The simplest is the first linear model, which is known as the LSS model. It requires two initial experiments to start the optimization, but sometimes it does not completely predict correctly the effect of concentration of an organic solvent in a MP. This can be observed for basic and acidic compounds that contain highly polar and charged structural fragments. Such fragments are typically observed in natural and pharmaceutical compounds, and retention models for such compounds are nonlinear in many cases. Additional experiments as a rule do not lead to improvement in the accuracy of the linear model when it is applied for nonlinear functions.

The quadratic model describes retention more adequately. Additional experiments improve the accuracy, but three initial experiments are required to start computer optimization. The higher the power of a model, the more complex retention behavior can be described and the more initial experiments must be performed to start optimization of separation.

ChromSword® supports optimizing separation for polynomial models up to power 6. A chromatographer optionally can choose from powers 1 to 6. Typically, the powers 1–3 are most commonly used; however, the most complex retention can be described and separation optimized with the higher polynomial powers.

All polynomial models predict the retention of solutes rather precisely in the interpolation region of those concentrations studied. These models are less reliable in the extrapolation region. For example, if experiments were performed with 40% and 50% of the organic solvent in a MP, one can expect rather a good prediction of retention and separation in the region between of these concentrations and less accuracy in the regions of 30–35% and 50–55%. Extrapolation within wider limits very often leads to substantial deviations between predicted and experimental data.

(B) An approach that takes into account both the features of solutes being separated and the characteristics of the stationary and MPs being used:

In this method, the two-layer continuum solvatic retention model was proposed [14, 15] as an extension of the solvophobic model of RPLC [16]:

- The surface of a modifier sorbent in RPLC has a surface layer that involves hydrocarbon radicals and some of the components of a MP.

- The surface layers are assumed as being quasi-liquid having their own physical characteristics i.e surface tension and dielectric permittivity.
- The surface characteristics vary with varying the MP composition and SP properties.
- Molecules of retained substances penetrate into the surface layer.
- The retention is determined by the difference in molecule solvation energies in the mobile and SPs.

In this model, the retention of a solute is derived as

$$\ln k = a(V)^{2/3} + b(\Delta G) + c \tag{4}$$

where V is the molecular volume of a solute, ΔG is the energy of interaction of a solute with water, and a, b and c are the parameters which are determined by the characteristics of a reversed-phase column in the eluent being used, i.e. surface tension, dielectric permittivity and others. This approach works more precisely and rapidly than that based on formal linear and quadratic polynomial models, but it requires that both the parameters of the solutes (volume and energy of interaction with water) and the characteristics of the reversed-phase column under experimental conditions be known.

The characteristics of different commercially available RPLC columns were experimentally determined initially in a wide range of concentrations of methanol and acetonitrile in water. ChromSword® contains a database of characteristics for more than 150 commercially available reversed-phase columns in these eluents; they load automatically when a column and an eluent are chosen from the software menu.

ChromSword® calculates the parameters of compounds from the structural formulae. If structural formulae of the compounds being studied is not known or a user decides not to draw them, these parameters can be determined by ChromSword® from the two chromatographic experiments with different concentrations of an organic solvent in a MP.

This approach enables ChromSword® to predict regular or irregular retention behavior of solutes separated and enables a chromatographer to move rapidly to achieve maximal separation in minimal time. Each additional experiment leads to an improvement in the predictability.

Thus, this approach enables a chromatographer to start optimizing retention without any preliminary tests if the structural formulae of the compounds are known and also enables one to start optimization of separation on entering the retention data for only one run.

For solutes with unknown or undefined structures, this approach can also be used after entering the retention data and chromatographic conditions for two runs.

The main advantage of the structure and column properties related approach is that it "fills" both a column and compound features. It works precisely in the interpolation region and reliably in the extrapolation region. Figure 3.12 and Table 3.2 show that the solvatic model provides a

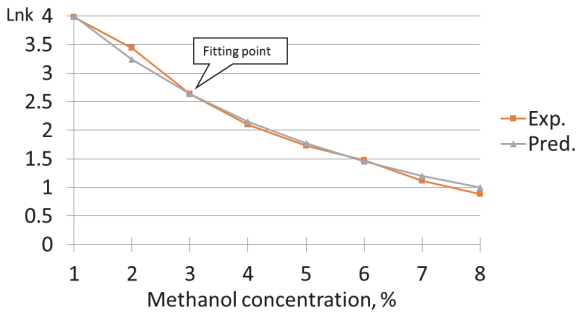

Figure 3.12: Adenosine monophosphate: predicted and experimental retention. Input: structure and data of one run at 3% MeOH. Column: Purospher RP-18e, 5 μm. MP: MeOH $-$ phosphate buffer, $\mathrm{pH} = 2.5$.

Table 3.2: Predicted and experimental retention of the beta-blocker carazolole in the extrapolated region of concentration of MeOH in a MP.

MeOH (%)	k_{exp}	K_{linear}	Dev (%)	$K_{\mathrm{quadratic}}$	Dev (%)	k_{Solvatic}	Dev (%)
60	4.62	4.62		4.62			4.62
50	6.33	6.33		6.33			6.33
45	8.83	7.71	-12.7	8.83		8.26	-0.33
30	33.57	19.70	-41.3	38.74	15.4	31.90	-4.97

Note: Retention values at 60% and 50% were used as input for the linear and solvatic models and at 60, 50 and 45% for the quadratic model. Column: Purospher RP 18e, 5 μm, 150×4 mm. MP: MeOH $-$ 50 mM phosphate buffer, $\mathrm{pH} = 3.5$.

good enough prediction of retention behavior for highly polar compounds that contain both uncharged and charged highly polar fragments.

3.3.3 *Procedure for optimizing pH in RPLC*

When a sample contains basic or acidic compounds with ionizable atoms or groups, pH is a very effective tool for optimizing the separation. ChromSword® supports two mathematical procedures for optimizing pH in RPLC. The first procedure is based on applying polynomials with powers up to 6 and the second procedure determines, using the retention data obtained with different pH values of a MP, the nature of solutes (neutral, acidic, basic), their p*Ka* value and then builds their retention models.

3.3.3.1 *Polynomial models*

The first three members are:

$$\ln k = a + b(pH) \tag{5}$$

$$\ln k = a + b(pH) + d(pH)^2 \tag{6}$$

$$\ln k = a + b(pH) + d(pH)^2 + e(pH)^3 \tag{7}$$

The powers 4–6 optionally can be employed for describing the most complex dependencies between retention and pH value of a mobile phase.

In order to optimize pH, a user must enter experimental retention data for two or more isocratic or gradient runs with different pH value of a MP. By analyzing retention data, ChromSword® determines and then refines the parameters of the retention model for the column being used and predicts the conditions for the best separation.

Tasks of a user are the same as that for optimizing separation in RPLC using a polynomial model and is described in Chapter 2 "procedure" for method development in HPLC using polynomial models.

3.3.3.2 *Fit pKa optimizing procedure*

This procedure determines, using the retention data obtained with different pH values of a MP, the nature of solutes (neutral, acidic, basic), their p*Ka* values and then builds their retention models:

$$k = k(0) + k(i)/(1 + F) \tag{8}$$

where $k(i)$ is the retention factor of an ionic form of a solute, $k(0)$ is the retention factor of a molecular form of a solute, and F is $Ka/[H^+]$ for acids and $[H^+]/Ka$ for bases, where *Ka* is the dissociation constant of a solute.

In order to optimize the pH value using the fit p*Ka* procedure, a user must enter experimental retention and efficiency data for three or more isocratic or gradient runs with different pH values of a mobile phase. By analyzing retention data, ChromSword® determines the nature of the compounds (base, acid, neutral) studied at pH intervals, calculates the p*Ka* values and then refines the parameters of the retention models for the column being used (Table 3.3, Fig. 3.13).

Substantial differences can be seen for retention time of basic and acidic compounds predicted by the p*Ka* and quadratic retention models. The p*Ka*-related model typically predicts retention for acidic and basic compounds better (Table 3.4).

Deviations in predicted retention can lead to a substantial difference in predicted optimal pH value for separation of a mixture with basic and acidic compounds. In Figs. 3.14 and 3.15, the resolution maps as functions of the quadratic and the fit p*Ka* models are shown for optimization of separation of a mixture of sweeteners and preservatives.

The Fit p*Ka* procedure enables a user to not only optimize the separation but also determine the nature of the compounds and evaluate their p*Ka*

Table 3.3: The p*Ka*-related model parameters determined for mixtures of nucleobases and nucleosides.

	Compound	Nature	k_0	k_i	*pKa*
1	Uracil	Neutral	1.12		
2	Cytosine	Base	0.78	0.51	5.63
3	Thymine	Neutral	3.77		
4	Uridine (U)	Neutral	3.10		
5	Cytidine (C)	Base	2.06	1.34	4.45
6	Ara-U	Neutral	4.43		
7	Ara-C	Base	2.68	1.68	4.17
8	6-azauridine	Acid	1.54	1.20	5.62
9	6-azacytidine	Neutral	0.98		
10	5-azacytidine	Base	2.21	1.47	4.04

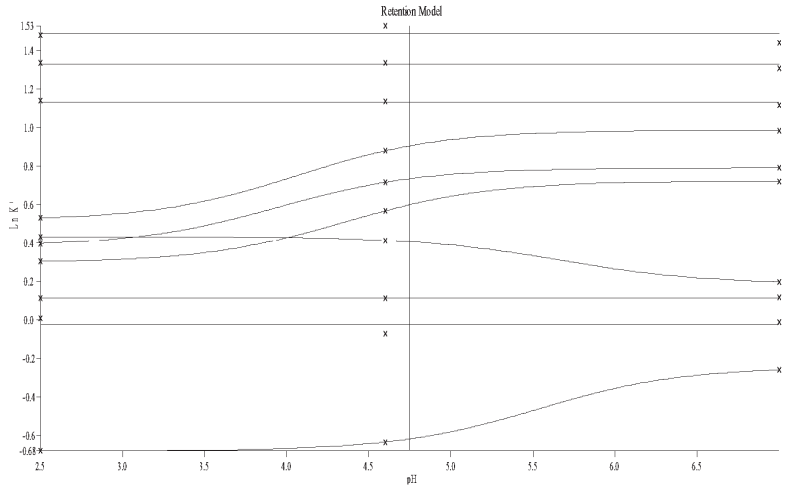

Figure 3.13: Retention models $(\ln \mathbf{k} = f(\mathbf{p}H))$ built with the Fit p*Ka* procedure for the compounds listed in Table 3.3 Column: Purospher RP18e, 5 μm, 125×4 mm. MP: 20 mM phosphate buffer $\mathrm{pH} = 2.5$; 4.6, 7.0. Flow rate 0.8 mL/min, $T = 35°$C.

Table 3.4: Predicted retention time with the quadratic ($\mathrm{RT_q}$) and p*Ka*-related ($\mathrm{RT_{pK}}$) model. $\mathrm{RT_e}$ — experimental values.

	Compound	$\mathrm{RT_q}$	$\mathrm{RT_{pK}}$	$\mathrm{RT_e}$	p*Ka*
1	Sorbic acid	7.54	10.00	10.00	4.67
2	Benzoic acid	4.76	5.41	5.37	4.19
3	Acesulfame	2.63	2.61	2.64	
4	Saccharine	3.41	3.45	3.43	
5	Aspartame	14.18	14.41	14.35	
6	Caffeine	7.07	7.06	7.08	

values under the conditions of a chromatographic experiment. In Tables 3.3 and 3.4, the p*Ka* values calculated from the experimental data are listed. It should be noted that the chromatographic method for the determination of p*Ka* values has advantages over other methods because it can be applied for mixtures and requires only a small amount of compounds.

It is necessary to take into account that the fit p*Ka* procedure assumes solutes to be monoprotic; therefore, for diprotic (and more) solutes as

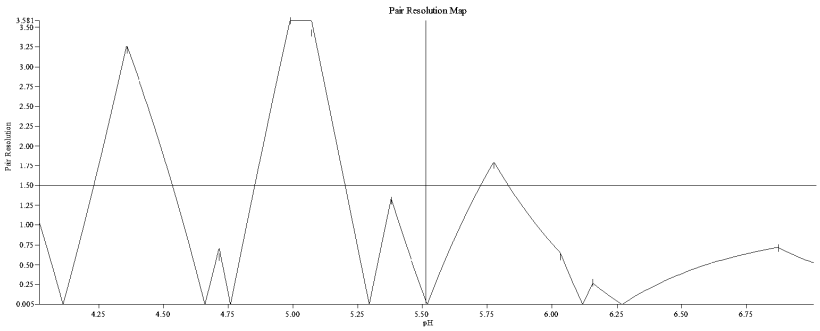

Figure 3.14: Resolution map built with the Fit p*Ka* procedure. Separation of the caffeine, acesulfame, saccharine and benzoic and sorbic acids. Column: Purospher RP18e, 5 μm, 125×4 mm. MP: 10% ACN/90% 20 mM phosphate buffer, $\mathrm{pH} = 7.01$; 4.02, 5.75. Flow rate = 0.8 mL/min, $T = 30^{\circ}$C.

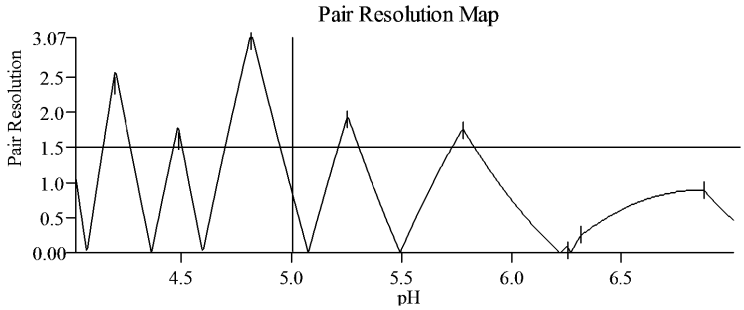

Figure 3.15: Resolution map built with the quadratic model. Conditions and mixture as described for Fig. 3.14.

well as for zwitterions, p*Ka* values can be considered as conditional. Nevertheless, this procedure can give valuable information about unknown compounds.

3.3.4 *Optimization of NPLC methods*

For optimization of the separation in NPLC, ChromSword® now supports only polynomial retention models. Retention in the NPLC can be described rather adequately by bilogarithmic models. ChromSword®

supports polynomials up to a power of 6. The first three are the following:

$$\ln k = a + b(\ln C) \tag{9}$$

$$\ln k = a + b(\ln C) + d(\ln C)^2 \tag{10}$$

$$\ln k = a + b(\ln C) + d(\ln C)^2 + e(\ln C)^3 \tag{11}$$

where C is the concentration of the stronger solvent in the mobile phase.

The powers 4–6 can be employed for describing the most complex dependencies between retention and concentration of a modifier in a MP.

In order to optimize a separation in NPLC, it is necessary to enter experimental retention and efficiency data for two or more runs with different concentrations of a strong solvent in the MP. By analyzing the retention data, ChromSword® determines and then refines the parameters of the retention model for a column being used and predicts the conditions for the best separation.

User tasks are the same as for optimizing separation in RPLC by using polynomial model and described in Chapter 2 "*procedure*" for method development in HPLC using polynomial models.

3.3.5 *Optimization of IEX methods*

The effect of the buffer concentration in the MP on retention in IEX can be described adequately by the same functions as for NPLC. Thus, a user can utilize the same procedure both for normal-phase and for IEXLC.

In order to optimize a separation in IEXLC, the user must enter experimental retention and efficiency data for two or more isocratic or gradient runs with different concentrations of a counter-ion in the MP. By analyzing the retention data, ChromSword® determines and then refines the parameters of the retention model elution for the column being used and predicts the conditions for the best separation.

3.3.6 *Optimization of the temperature*

Optimizing the temperature can be an effective tool if the conformation of solutes changes with temperature. This phenomenon can be observed

rather often in the case of large molecules such as peptides, proteins or for molecules with bulky substituents. In general, the effect of temperature on the logarithmic retention factor can be described by the simple equation $\ln k = a + b(1/T)$ for any mode of chromatography including gas chromatography. But if a solute changes its conformation, the function $\ln k = f(1/T)$ can be much more complex.

To optimize the temperature of a chromatographic separation, ChromSword® uses up to six power polynomials. The first three are the following:

$$\ln k = a + b(1/T) \tag{12}$$

$$\ln k = a + b(1/T) + d(1/T)^2 \tag{13}$$

$$\ln k = a + b(1/T) + d(1/T)^2 + e(1/T)^3 \tag{14}$$

where T is the temperature of the MP.

For optimizing the temperature, the same procedure as for optimizing the concentration of a modifier in RPLC, NPLC and IEX can be used.

In order to optimize a separation, the user must enter experimental retention and efficiency data for two or more runs with different temperatures of the MP. By analyzing the retention data, ChromSword® determines and then refines the parameters of the retention model elution for the column being used and predicts the conditions for the best separation.

If the model with the power one is applied, then ChromSword® also determines the enthalpy of sorption from the retention model:

$$\ln k = \ln k_0 + \Delta H/(RT) \tag{15}$$

where ΔH is the enthalpy of sorption of a solute in kJ/mol and R is the universal gas constant.

Thus, ChromSword® can be applied not only for optimizing a separation but for physico-chemical studies of compounds. For unknown compounds, ΔH values can be useful for elucidation of their structure.

3.3.7 *Optimization of the gradient*

There are different approaches to optimize gradient profiles after the determination of the retention models. The most frequently used approach is the

optimization of linear gradient profiles when two runs with linear gradients and different gradient times are used as input. These runs are used to build retention models. The initial and final concentrations of a modifier are fixed both for input and optimization. In this case, only the gradient time is optimized. This is simple approach that can easily be combined with the optimization of other variable like the temperature of a MP. However, complex mixtures in many cases can be separated only with multi-step gradient profiles. These include natural samples or samples after force degradation tests in pharmaceutical research and development laboratories. Every gradient node can be characterized by two parameters — time and concentration — and the position of every node in the time and the concentration dimensions should be optimized. Such multi-step gradients can be optimized by simulating chromatograms for different multi-step gradient profiles; however, this is not a fast method.

To build retention models, ChromSword® can process two or more runs with linear or (and) multi-step gradients. In this case, every new run can be used to refine retention models. For the optimization of both linear and multi-segment gradient profiles, the Monte Carlo and genetic algorithms are used. A user needs to enter the parameters of optimization, desired run time, separation and target peaks to be separated, and the stochastic procedure will find the best gradient profile automatically, assuming the separation is possible. The more segments on the gradient profile and compounds in a sample, the more time for optimizing is necessary. Typically, ChromSword® spends only a few minutes with conventional PCs finding the best multi-segmented gradient profile.

3.3.8 *Optimizing two variables simultaneously*

Optimization of two variables is an effective tool for improving and developing HPLC methods. ChromSword® provides all necessary interface and mathematical procedures for optimization of two chromatographic variables simultaneously. The following two variables can be optimized with ChromSword®:

Using one column:

- gradient profile and temperature
- concentration of a modifier in a MP and temperature

- pH and temperature
- concentration of an organic solvent and pH
- concentration of two different organic solvents

Using up to four connected columns with different selectivity (*column coupling, column combination*):

- gradient profile and ratio of columns
- concentration of an organic modifier and column ratio for RPLC
- concentration of an organic modifier and column ratio for NPLC
- pH and column ratio for RPLC
- temperature and column ratio for RPLC and NPLC

3.3.9 *Simultaneous optimization of a gradient profile and temperature*

Gradient and temperature optimization procedure allows the user to predict retention and to optimize the separation in gradient elution by entering retention data and experimental conditions for three or more gradient runs with different slopes and temperature. It is practically useful that the gradient profiles can be both linear and multi-step. One of the possible plans of the experiments can be for a user to perform two linear gradients with different slopes and same temperature and the third linear gradient with a different temperature. The slopes should be substantially different.

- Run 1: 20 min linear gradient with concentration of an organic solvent ranging from 5% to 95% at temperature 30°C.
- Run 2: 40 min linear gradient with concentration of an organic solvent ranging from 5% to 95% at temperature 30°C.
- Run 3: 40 min linear gradient with concentration of an organic solvent ranging from 5% to 95% at temperature 40°C.

The difference in temperature should be, in the majority cases, not less than 10°C between gradients.

When the user inputs data of experimental runs (retention, efficiency, area) and conditions (gradient profiles, temperature, column dead time, the dwell time of the HPLC system), ChromSword® builds retention models and the user can compute simulate experiments with different profiles

and temperatures. It is also possible to search for optimal gradient profile and temperature using the automatic procedure. The simplest approach that is used in different method development software is to build resolution maps where the resolution is a function of the gradient time and temperature. In this case, the initial and final gradient time values are fixed and cannot be optimized automatically. The user should change the initial and final MP compositions and observe their impact on the resolution map. This manual procedure takes substantial time, even for simple linear gradients. For example, to study the effect of initial and final concentrations for $+/-5\%$ it is necessary to simulate 100 resolution maps for all combinations of the initial and final concentration. For multi-step gradients, the number of computer experiments to simulate the position of every gradient point and their combinations is enormous. Automated optimization procedures that are implemented in ChromSword® have no such limitations and enable a user to optimize simultaneously the initial and final concentrations, gradient time and temperature for linear gradient profiles or the temperature and position of all nodes in multi-step gradient profiles.

When a user finds a promising gradient profile with ChromSword® and performs the run, it is also possible to input the obtained retention data to refine retention models and then repeat the computer simulation and optimization.

3.3.10 *Optimization of separation using supervised machine learning*

In recent years, machine learning-based models have been able to solve problems that previously could be resolved only by experts [17, 18]. Deep machine learning models on limited datasets were applied for the prediction of retention time of peptides in RPLC [11]. In earlier publications, outdated artificial neural network methods were utilized to predict retention time of simple samples and a few linear gradients [19, 20]. None of these contributions attempted at finding multi-step solvent gradient for separation of compounds. We applied machine learning as one of the optimization methods in ChromSword®. The deep machine learning technology was not utilized widely in chromatography. We consider that some

information on its possibility in gradient optimization can be interesting for both computer scientists and specialists in computer-assisted method development.

For the deep learning model, we used the recurrent neural network (RNN). An efficient algorithm for RNN is the long short-term memory (LSTM) cells [21]. The LSTM cell in an RNN-based model is a recursive function that uses a set of sub-functions. This function receives input data from the training set for every time step. In our case, the time steps are training runs in a sequence of runs. Then, this function tries to forecast the desired result as an optimal solvent gradient to achieve good separation of compounds. Parameters of sub-functions inside the LSTM cell are trained using modern variations of stochastic gradient descent (SGD) algorithms. It should be noted that the LSTM cannot be applied directly to produce usable method conditions because the resulting value will be in a range from -1 to 1 (*tanh* function). To use LSTM layers, we need to normalize input and output data vector to appropriate scale or to use as a last layer linear regression of deep learning model. The linear regression layer would then produce usable values for concentration in a range between 0 and 100. As for the input part of the LSTM, we use the convolutional neural network (ConvNet) [22] to embed features of scouting runs like data points of the chromatogram, spectra, retention time of compounds, solvent concentration gradient, temperature, etc. A very promising development in machine learning research in recent years has been made in the field of deep reinforcement learning [23]. These algorithms use a model that learns regression task when it tries to forecast the cumulative reward of the whole trajectory of actions to perform a predefined task. It means that we can train a model to generate method conditions for a sequence of runs that will gradually lead to the best separation of compounds. For each run, the quality of the result (reward) can be estimated using the sum of pair resolution values for each peak in a run. The model calculates cumulative reward value for each run in a sequence. Using these rewards, the model learns to construct a gradient, extract knowledge from the acquired chromatogram and then construct the next gradient that will have a higher reward value

$$R = \sum_{t=0}^{n} \gamma^t r_t \tag{16}$$

Cumulative reward R is calculated by summing all rewards r_t of runs multiplied by discount constant γ^t that reduces the importance of future rewards at the present state. We cannot use R value directly to train our model, because it takes into account only executed actions. For example, it calculates rewards from runs with method conditions in a specific sequence, but we would like to construct utility function to train our model to include more possible method conditions. To realize this, we can construct the quality method value (Q)-based model using Bellman's equation Q_π that takes advantage of partial Markov decision process property:

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha\Big(r_t + \max_a Q_\pi(s_{t+1}, s_{a+1}) - Q_\pi(s_t, a_t)\Big) \tag{17}$$

State s_t contains retention time, width of peaks, pair resolutions and other important method quality characteristics. Action a_t contains proposed a concentration gradient and other method conditions. We try to maximize Q-value that is approximated cumulative reward by changing method conditions.

To train the deep reinforcement learning model, we used physical retention models generated by ChromSword® as a training environment. The retention models were determined from retention behavior of different families of compounds, like small molecules and proteins. Then, a special procedure generated a large dataset of runs and simulated chromatograms for the training. In fact, the pattern of chromatograms as a function of solvent gradients and other conditions like temperature or pH can be used for the training. When beginning the training set, the Q-value model produces random method conditions; however, after training — using distributed computing — it can be applied to new samples. Our results showed that after training with simulated samples, the procedure can process the results of scouting runs of real samples and predict gradient profiles to provide a reasonable separation.

3.3.11 *Column coupling*

ChromSword® provides support in the case of the most complex mixtures when no acceptable conditions were found with several types of columns. In this case, the chromatographer can try to separate a mixture by coupling

columns with different selectivities. To optimize separation on coupled columns, it is possible to use data that were obtained separately for different single columns. Typically, columns with 2, 5, 7.5, 10, 15 and 25 cm lengths are commercially available and can be easily combined by using dead volume connectors or column cartridges. In this case, the generic procedure can be applied. This is done as follows:

- Make several runs with different concentrations of a modifier or gradient profiles in column 1.
- Input data of the runs for the column 1 page.
- Build retention models for compounds being separated.
- Build the pair resolution map, search for promising regions and simulate chromatograms.
- If no acceptable conditions are found, a user has choice for the next step:

  - Try an other type of column (columns 2, 3, 4).
  - Try an other solvent and pH or/and temperature with column 1.

If the chromatographer chooses the first option (change a column), it is possible to repeat the same steps 1–4 to try to optimize the concentration of a modifier in the MP or the gradient profile with that of the column 2. The other conditions must be the same as used for column 1 (solvent type, temperature, pH). If no good separation was found with column 2, the user can perform a computer simulation on:

- Coupling of columns 1 and 2 (a maximum of four columns can be virtually coupled) and optimizing the ratio of column lengths or column segments.
- Effect of the concentration of organic modifiers or the gradient profile on the separation for coupled columns.

The same procedure can be used for optimizations of pH or temperature and column coupling simultaneously.

3.4 Conclusion

ChromSwordAuto® is a software package which includes a chromatography method development data system and ChromSword® module for off-line computer-assisted method development.

ChromSwordAuto® is used for automatic method development of small and large molecules and supports mechanistic and statistic approaches for the optimization of method variables. ChromSwordAuto® also contains a module for high-throughput screening of many SP and MP combinations.

ChromSword® and ChromSwordAuto® are used for method development and optimization in practically all types of LC.

References

[1] S.V. Galushko, A.A. Kamenchuk, G.L. Pit, The calculation of retention and selectivity in RP LC. IV. Software for selection of initial conditions and for simulating chromatographic behaviour, *J. Chromatogr.* 660 (1994) 47–59.

[2] S.V. Galushko, V. Tanchuk, I. Shishkina, O. Pylypchenko, W.D. Beinert, ChromSword® software for automated and computer-assisted development of HPLC methods, In *HPLC Made to Measure: A Practical Handbook for Optimization*, ed. Stavros Kromidas, WILEY-VCH Verlag GmbH & Co. KgaA, 2006, pp. 557–570.

[3] Industry Analytical Procedures and Methods Validation for Drugs and Biologics. https://www.fda.gov/downloads/drugs/guidances/ucm386366.pdf.

[4] E. Hewitt, P. Lukulay, Implementation of a rapid and automated high performance liquid chromatography method development strategy for pharmaceutical drug candidates, *J. Chromatogr. A* 1107 (2006) 79–87.

[5] K.P. Xiao, Y. Xiong, F.Z. Liu, A.M. Rustum, Efficient method development strategy for challenging separation of pharmaceutical molecules using advanced chromatographic technologies, *J. Chromatogr. A* 1163 (2007) 145–156.

[6] S. Larson, G. Gunawardana, M. Preigh, Automated method development in HPLC. Evaluation of the ChromSword® software package. HPLC 2007 Abstract book, P23.06. http://www.chromatographyonline.com/efficient-chiral-hplc-method-development-using-chromsword-software.

[7] F. Vogel, S.V. Galushko, Automated development of reversed-phase HPLC methods for separation of chiral compounds, *Chromatogr. Today* 8 (2015) 54–55.

[8] L.W. Snyder J.W. Dolan, *High Performance Gradient Elution*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2007, p. 228.

[9] S. Fekete, S. Rudaz, J. Fekete, D. Guillarme, Analysis of recombinant monoclonal antibodies by RPLC: Toward a generic method development approach, *J. Pharm. Biomed. Anal.* 70 (2012) 158–168.

[10] S. Fekete, A. Beck, J. Fekete, D. Guillarme, Method development for the separation of monoclonal antibodycharge variants in cation exchange chromatography, Part I: Saltgradient approach, *J. Pharm. Biomed. Anal.* 102 (2015) 33–44.
[11] J. Koyama, J. Nomura, Y. Shiojima, Y. Ohtsu, I. Horii, Effect of column length and elution mechanism on the separation of proteins by reversed-phase high performance liquid chromatography, *J. Chromatogr.* 625 (1992) 217–222.
[12] ICH Topic Q 2 (R1) "Validation of Analytical Procedures"
[13] http://www.smatrix.com/products.html.
[14] S.V. Galushko, The calculation of retention and selectivity in RPLC, *J. Chromatogr.* 552 (1991) 91–102.
[15] S.V. Galushko, The calculation of retention and selectivity in RPLC. II. Methanol–water eluents, *Chromatographia* 36 (1993) 39–41.
[16] Cs. Horvath, W. Melander I. Molnar, Solvophobic interaction in liquid chromatography with nonpolar stationary phases, *J. Chromatogr.* 125 (1976) 129–140.
[17] M. Ren, R. Kiros, R.S. Zemel, Exploring models and data for image question answering, arXiv:1505.02074.
[18] J. Donahue, L.A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, T. Darrell, Long-term recurrent convolutional networks for visual recognition and description, arXiv:1411.4389.
[19] N.H. Tran, X. Zhang, L. Xin, B. Shan, M. Li, De novo peptide sequencing by deep learning, *PNAS* 114 (2017) 8247–8252.
[20] T. Bolanča,Š. Cerjan-Stefanović, M. Novč, Application of artificial neural network and multiple linear regression retention models for optimization of separation in ion chromatography by using several criteria functions, *Chromatographia* 61 (2005) 181–187.
[21] H. Wang, W. Liu, Optimization of a high-performance liquid chromatography system by artificial neural networks for separation and determination of antioxidants, *J. Sep. Sci.* 27 (2004) 1189–1194.
[22] Y. Li, Deep reinforcement learning: An Overview, http://arxiv.org/abs/1701.07274.
[23] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, A brief survey of deep reinforcement learning, arXiv:1708.05866.

APPENDIX C - Paper 3

Survey of Deep Q-Network variants in PyGame Learning Environment

Evalds Urtans
Riga Technical University
Kalku iela 1
Riga, Latvia
+371 26401317
evalds.urtans@rtu.lv

Agris Nikitenko
Riga Technical University
Kalku iela 1
Riga, Latvia
+371 26401317
agris.nikitenko@rtu.lv

ABSTRACT
Q-value function models based on variations of Deep Q-Network (DQN) have shown good results in many virtual environments. In this paper, over 30 sub-algorithms were surveyed that influence the performance of DQN variants. Important stability and repeatability aspects of state of art Deep Reinforcement Learning algorithms were found. Multi Deep Q-Network (MDQN) as a generalization of popular Double Deep Q-Network (DDQN) algorithm was developed. Visual representations of a learning process as Q-Value maps were produced using PyGame Learning Environment. Videos of trained models available in following link: http://yellowrobot.xyz/mdqn

CCS Concepts
• **Theory of computation → Design and analysis of algorithms**
• **Applied computing**

Keywords
Deep Reinforcement Learning; Deep Learning; DQN; DDQN; MDQN.

1. INTRODUCTION
This paper surveys many of the latest Deep Q-Learning algorithms in the field of Deep Reinforcement Learning. Notable examples of Deep Q-Learning (DQN) algorithms are the original DQN [13], Double Deep Q-Learning (DDQN) [5], Dueling Network [23] and asynchronous n-step DQN [14]. In addition to these Q-Value based algorithms, there are two other major branches of development in this field. One is policy gradient methods, with notable algorithms like Trust Policy Region Optimization (TRPO) [17] and Proximal Policy Optimization [19].

Another branch is a combination of Q-Value and policy gradient models that are called actor-critic model with notable algorithms like Deep Deterministic Policy Gradient (DDPG) [12], Asynchronous Advantage Actor-Critic (A3C) [14], GPU A3C [2] and Actor-Critic with Experience Replay (ACER) [22]. This paper focuses only on Q-Value function based algorithms.

ICDLT '18, June 27–29, 2018, Chongqing, China
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6473-7/18/06…$15.00
https://doi.org/10.1145/3234804.3234816

The paper explores how variations of these algorithms and hyper-parameters affect performance in PyGame Learning Environment (PLE) [21]. In this paper generalization of the DDQN algorithm and extension is proposed. It can use 2, 3 or more decoupled DQN models.

2. RELATED WORK
Recently some surveys have been conducted to assess a huge variety of Deep Reinforcement Learning algorithms [11], [3], [9].

Many Deep Reinforcement Learning algorithms suffer from large variance in results. There have been a number of papers trying to resolve this issue [1], [18].

Some research also points out problems with repeatability and identifies random seed as a significant factor that impacts results [10], [8].

3. METHODOLOGY
3.1 Deep Q-Network variants
All Q-function algorithms share underlying equations: the calculation of Cumulative Reward Equation 1 and the Bellman equation for modeling policy π through Q-function Equation 2 that is an approximation of a reward function for a given state s and action a at a time step t.

$$R = \sum_{t=0}^{n} \gamma^t r_t \tag{1}$$

$$Q_\pi(s_t, a_t) = r_t + \max_{a'} Q_\pi(s_{t+1}, a') \tag{2}$$

DQN algorithm relies on Equation 3 where a parametrized Q-function is based on a deep neural network. Usually, an input is a raw pixel representation trained by Convolution Neural Network (ConvNet/CNN) or a lower dimensionality representation of s. The model also usually utilizes Recurrent Neural Network (RNN) like LSTM or GRU.

$$Q_\Theta(s_t, a_t) \leftarrow Q_\Theta(s_t, a_t) + \alpha(\nabla((r_t + \max_{a'} Q_\Theta(s_{t+1}, a') - Q_\Theta(s_t, a_t)))) \tag{3}$$

$$Q_\Theta(s_t, a_t) \leftarrow Q_\Theta(s_t, a_t) + \alpha(\nabla((r_t + \max_{a'} Q_{target}(s_{t+1}, a') - Q_\Theta(s_t, a_t)))) \tag{4}$$

DDQN algorithm is similar to DQN, but it utilizes theory from Double Q-Learning [7] by using two decoupled Q-functions like shown in Equation 4. Q_{target} function parameters are copied from

Q_Θ with a given time step interval thereby achieving two decoupled Q-functions.

3.2 Multiple Deep Q-Networks

There are some differences of DDQN (Double Deep Q-Network via Target network) [6] and original DQL (Double Q-Learning) [7]. In case of DDQN Q_{target} is used as decoupled function whereas in pure DQL there should be Q_1 and Q_2 that are used intermittently. DDQN is simpler and should preserve same properties as pure DQL.

The paper explores this simplification impacts performance and implemented a pure version of DDQN and compared it with a standard DDQN. DQL algorithm was generalized to use any number of decoupled functions in Bellman equation and call it MDQN (Multiple Deep Q-Network). MDQN with 2 decoupled functions is listed in Algorithm 1, but this could be easily expendable to more decoupled function pairs.

Algorithm 1: MDQN (2 decoupled functions)

procedure Train
  while Training == True **do**
    if random(0.0, 1.0) < 0.5 **then**
      if $s_t \neq$ terminal state **then**
        $Q_1(a_t, s_t) \leftarrow R_t + \gamma \max_a Q_2(a, s_{t+1})$
      else
        $Q_1(a_t, s_t) \leftarrow R_t$
    else
      if $s_t \neq$ terminal state **then**
        $Q_2(a_t, s_t) \leftarrow R_t + \gamma \max_a Q_1(a, s_{t+1})$
      else
        $Q_2(a_t, s_t) \leftarrow R_t$
    $a_t \leftarrow max_a average(\{Q_1(a, s_t), Q_2(a, s_t)\})$
    …

3.3 Other algorithmic improvements

Some algorithmic improvements have been made that can be applied to other deep reinforcement learning algorithms.

One of the improvements was to use the cumulative reward for training actions that were observed in an offline rollout of a episode. For example, if the offline state contains $\{s_t, a_t\}$ and calculated cumulative reward for $\{s_{t+1}, a_{t+1}\}$ then it is possible to train the model using cumulative reward value instead of Bellman equation. And when $\{s_t, a_t, s_{t+1}, a_{t+1}\}$ is not observed in an episode it is possible to use a value from Bellman equation. Principle is shown in in Algorithm 2.

In this research, RNN (Recurrent Neural Networks) were used as models of DQN variants. These models take as input observation from 5 previous frames. To speed up training, RNN-ReLU were used instead of LSTM or GRU. LSTM and GRU perform better than RNN-ReLU, but also take up to 7 times longer to train. Label smoothing were implemented to prevent vanishing gradients in RNN-ReLU [15].

All source code used to test algorithms in this paper is open-source. Prioritized replay buffer is implemented as a separate library that can be used with a completely different set of reinforcement learning algorithms[1]. It includes both types of prioritized replay buffer algorithms: proportional and ranked [16].

The main part of source code that contains variants of algorithms that were tested and is also available as an open-source project[2]. Code was implemented it in a way that it could utilize High-Performance Cluster (HPC) architecture. Every training using sample of random seed were executed as a separate task on a node in a cluster. Each sample of random seed was a complete training of 10^7 frames with specified hyper-parameters.

Algorithm 2: MDQN with a cumulative reward boost

procedure Train
  while Training == True **do**
    // Offline variant of an algorithm
    while $s_t \neq$ terminal state **do**
      $a_t \leftarrow max_a average(\{Q_1(a, s_t), Q_2(a, s_t)\})$
      …
      store $\{a_t, s_t, s_{t+1}, r_t\}$ in $ReplayBuffer$

    for $\{a_t, s_t, s_{t+1}\}$ sample from $ReplayBuffer$ **do**
      $a'_t \leftarrow max_a average(\{Q_1(a, s_t), Q_2(a, s_t)\})$
      if $\{a'_t, s_t, s_{t+1}\}$ in $ReplayBuffer$ **then**
        if random(0.0, 1.0) < 0.5 **then**
          $Q_1(a_t, s_t) \leftarrow \sum_{t=0}^{t+1} \gamma^t R_t$
        else
          $Q_2(a_t, s_t) \leftarrow \sum_{t=0}^{t+1} \gamma^t R_t$
      else
        if random(0.0, 1.0) < 0.5 **then**
          if $s_t \neq$ terminal state **then**
            $Q_1(a_t, s_t) \leftarrow R_t + \gamma \max_a Q_2(a, s_{t+1})$
          else
            $Q_1(a_t, s_t) \leftarrow R_t$
        else
          if $s_t \neq$ terminal state **then**
            $Q_2(a_t, s_t) \leftarrow R_t + \gamma \max_a Q_1(a, s_{t+1})$
          else
            $Q_2(a_t, s_t) \leftarrow R_t$

4. EXPERIMENTS

4.1 PyGame Learning Environment

In this research to evaluate results, an open-source game environments "PyGame Learning Environment" (PLE)[3] were used.

PLE contains many different games including Flappy Bird, 3D Maze, Doom, and others. For most of the game environments it is possible to get low dimensional representations of a state, which are useful for testing deep reinforcement algorithms with limited computational resources. Of course, it is also possible to train agents using high dimensional pixel representations of a state. Another very desirable feature is that game environments can be manipulated while running because full source code for each game is easily accessible.

Curriculum learning were implemented for the 3D raycast maze, where target moves away from starting point in later stages of training. Method to produce Q-value map (Q-map) were

[1] https://github.com/evaldsurtans/dqn-prioritized-experience-replay

[2] https://bitbucket.org/evaldsurtans/dqn-research

[3] https://github.com/ntasfi/PyGame-Learning-Environment

implemented by manipulating a position of a game character in an environment and getting Q-value for every artificial state in a game. For example, in a game of flappy bird, the bird character is moved across all pixels in a frame and a Q-function value is calculated that is overlaid as a heat map like in Figure 1. This kind of representation helps to understand what DQN model has learned. In fact, we found and fixed a bug in a Flappy Bird environment by using Q-map when we noticed that DQN model learned to cross an obstacle over the top of the screen. In case of 3D raycast maze, we implemented Q-map by teleporting a player to all walkable squares and rotating incrementally player's camera around the center of each square. For every frame, it is possible to calculate average Q-Value of all actions available and then make a heat map of a maze like in Figure 2.

4.2 Random seed and repeatability

Our research highlights a problem that all DQN, DDQN and MDQN variants are very sensitive to seed randomization. In this research method to restore all random seeds and repeat results were implemented, but this is not desirable because it can lead to misleading results when comparing different hyper-parameters. A better approach is to increase the sample size of random seeds. This means that every training configuration should be rerun multiple times with different randomization seeds as shown in Figure 3.

Large variance between different samples of random seed were observed. To make accurate comparisons, it is necessary to choose a random seed size of 10, since we observed that this resulted in similar variances to sample sizes 20 and 40. Whereas using a sample size of 5 produced a much lower variance of results.

To complete this research, we had quite limited computing resources and even random seed size of 10 took considerable time to test. It is one of the reasons why we chose experimentally initial hyper-parameter values that we changed one by one, instead of performing full grid search.

Often it is advised to reduce variance by reducing the model complexity [4]. Our results confirm this hypothesis Figure 4, however by reducing model complexity also a maximal average score of testing set reduces as well. When constructing such models, it is necessary to find the compromise between model complexity, repeated random seed test set size and a variance.

Another widely used method to reduce variance is to use regularization. Again, our results confirm that it reduces variance, but again it also reduces average scores as shown in Figure 5.

As for batch normalization, no significant improvement was found as shown in results in an appendix.

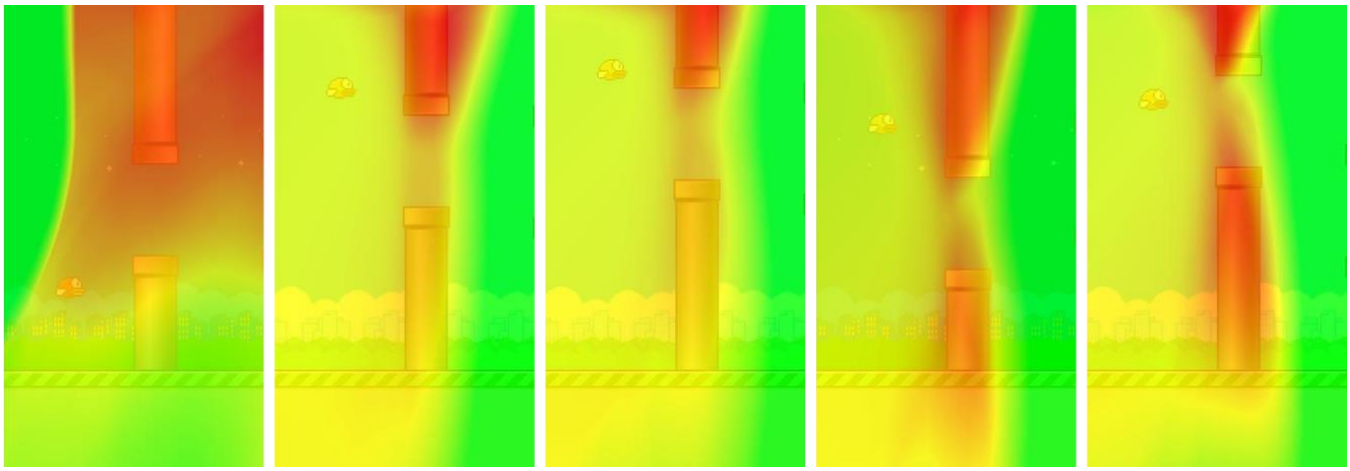

Figure 1. Q-Maps of sequential training of Flappy Bird from first frame on left till 10^7 frame on right. Green is highest value state. Red is lowest value state.

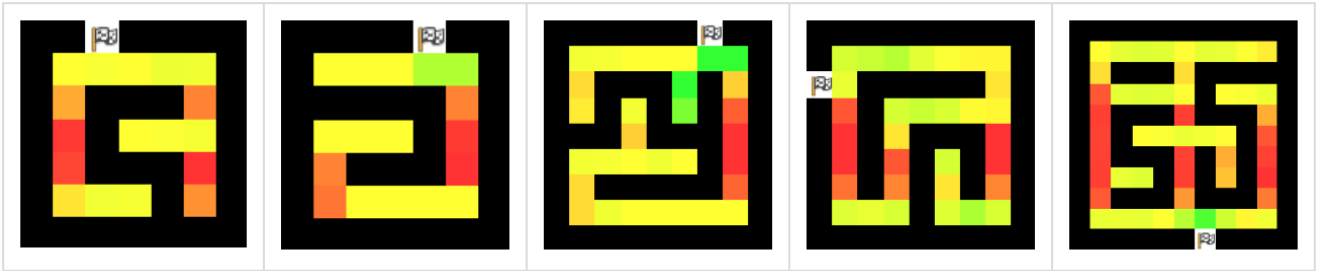

Figure 2. 3D Raycast maze Q-map for each position in map from top down view. Each Q-map represents sequential frame checkpoint during training. On left first frame and on right 10^7 frame. Notice that map increases in size thus using curriculum learning principle.

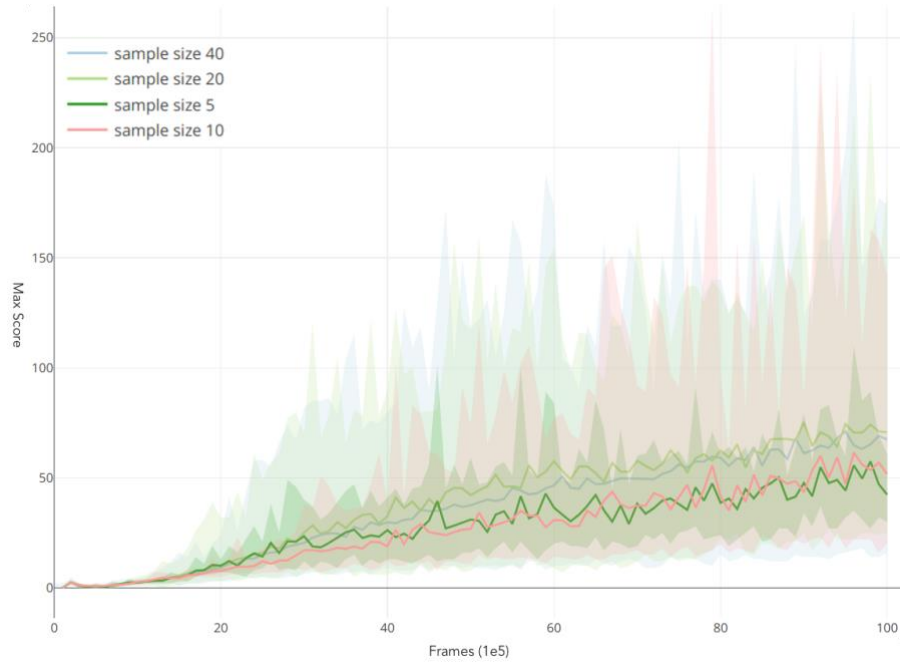

Figure 3. Sample size of random seeds and variance of average score for Flappy Bird environment.

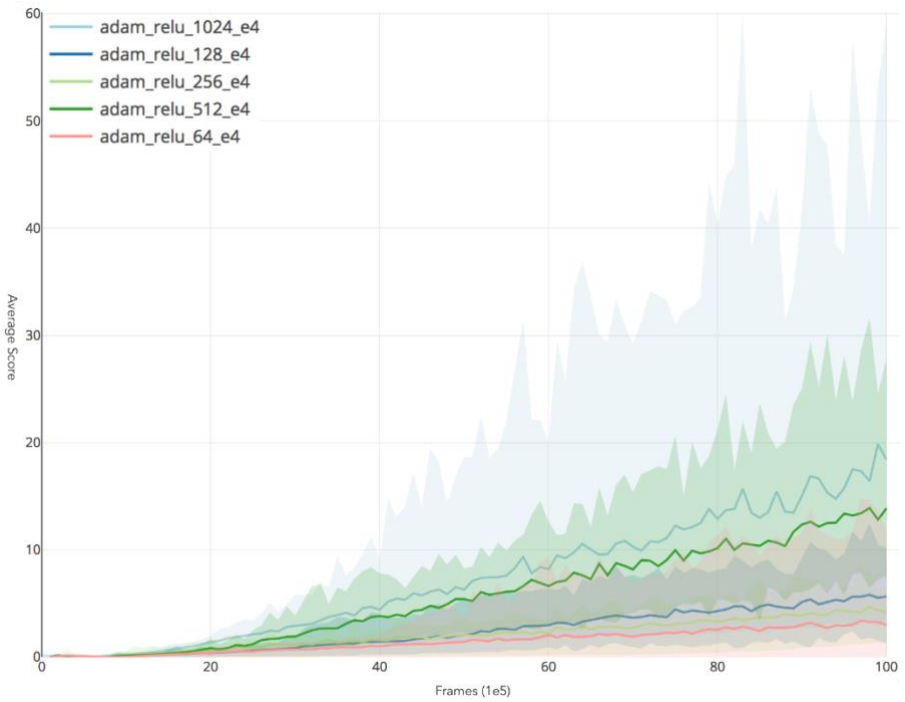

Figure 4. Comparison of different hidden unit vector sizes and variance of average score for Flappy Bird environment.

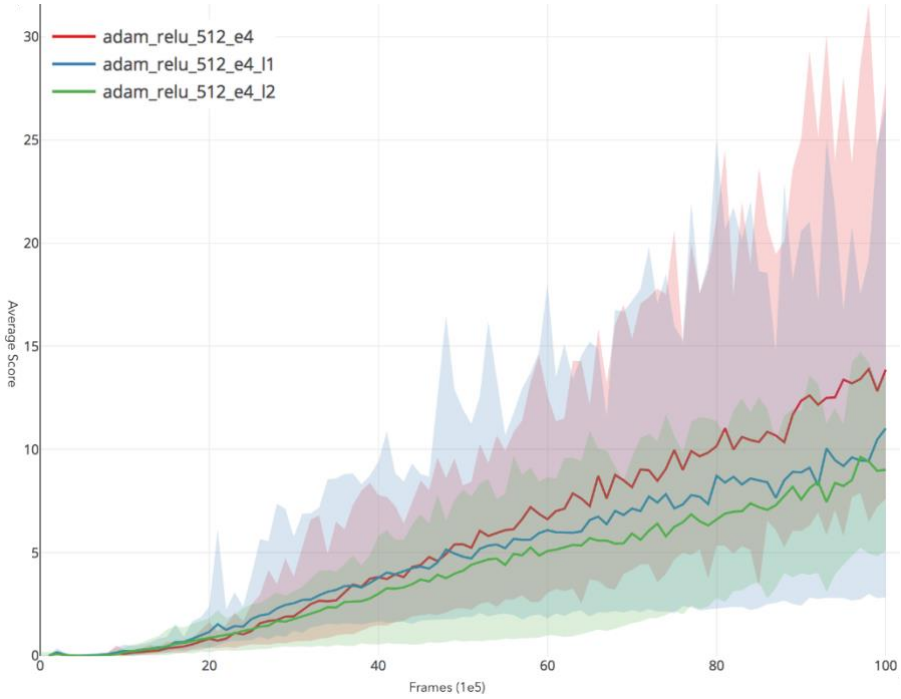

Figure 5. Effect of L1 (Lasso) and L2 (Ridge) regularization on variance of average score for Flappy Bird environment.

Table 1. Default hyper-parameters that other parameters were measured against in all environments

parameters	
batch norm: false	mini-batch: 32
bellman gamma: 0.99	model: 1 states to n actions
beta replay buffer: true	offline prebatch: false
cumulative reward: true	online: false
diff. states: false	optimizer: rmsprop
dropout: 0.0	pixels input: none
dueling arch.: false	priority replay buffer: ranked
epsilon greedy: true	reg.: none
epsilon start-end: 1e-3 - 1e-6	replay buffer: 5e5
epsilon stuck: false	rnn: relu
extra frame reward: 1e-5	sarsa: false
frames back: 5	state prev. act. reward: false
frames before: 5e4	target network alpha: 1.0
grad clip.: 0.0	terminal reward: -1e3

Table 2. Top 15 hyper-parameters of DDQN for Flappy Bird environment

parameter	lr	avg. score	max. score	var. score	time (min.)
rnn: gru	0.0001	42.97024986	264.1	549	2603.809
rnn: lstm	0.0001	28.60916534	264.1	99.2	3246.475
optimizer: adam	0.001	16.96737049	264.1	36.9	351.827
grad clip.: 1.0	0.001	12.45431387	254.096	19	399.956
optimizer: adam	0.0001	10.19414131	207.07831	21.7	367.396
grad clip.: 10.0	0.0001	9.765907544	184.06991	19.2	361.19
grad clip.: 10.0	0.001	8.694875819	156.05916	12.8	364.509
grad clip.: 1.0	0.0001	7.633139692	140.05331	17.1	357.085
rnn: lstm	0.001	6.614351436	182.06913	21.5	3770.588
optimizer: adam	0.00001	1.929185929	47.0181	1.81	375.648
mini-batch: 8	0.00001	1.922057009	40.0155	0.414	505.778
bellman gamma: 0.90	0.00001	1.916720841	47.01799	1.19	366.701
grad clip.: 10.0	0.00001	1.783358575	32.01237	0.322	400.212
beta replay buffer: false	0.00001	1.647591408	60.02323	1.02	423.562
default	0.00001	1.533690858	32.01236	0.308	377.689

Table 3. Top 15 hyper-parameters of MDQN3 for Flappy Bird environment

parameter	lr	avg. score	max. score	var. score	time (min.)
rnn: gru	0.0001	24.6588361	264.1	70.62906959	2735.039
rnn: lstm	0.0001	16.17153479	224.08495	31.69684243	3041.107
optimizer: adam	0.001	12.14972485	148.05634	8.572064894	357.792
rnn: lstm	0.001	6.362237775	161.06122	10.33771355	3052.423
grad clip.: 10.0	0.001	6.148186995	130.0494	11.12354631	380.245
grad clip.: 10.0	0.0001	5.774537436	104.03959	11.92569845	361.402
optimizer: adam	0.0001	5.541812022	124.0473	18.46937444	379.416
grad clip.: 1.0	0.0001	4.706386259	119.04543	11.58234856	374.33
grad clip.: 1.0	0.001	2.676778874	65.02486	4.486235409	456.614
rnn: gru	0.001	1.117617436	64.02454	1.294722902	2537.775
target network alpha: 0.0	0.00001	1.047481234	19.00759	0.298771101	745.303
model: n states to n act.	0.00001	0.881531711	12.00483	0.101958401	600.168
epsilon stuck: true	0.00001	0.878962391	11.00452	0.013849234	384.174
grad clip.: 10.0	0.00001	0.86220963	21.00828	0.183416314	402.766
mdqn: min	0.00001	0.740644674	10.0041	0.05608085	376.083

Table 4. Comparison of DQN, DDQN and MDQN models for Flappy Bird environment. Decimal number after abbreviation like mdqn3 1.0 denotes coefficient of target network. Coefficient 0.0 denotes that algorithm do not use target network.

model type	lr	avg. score	max. score	var. score	time (min.)
dqn 1.0	0.001	28.78679352	264.1	303.0517411	500.211
mdqn2 1.0	0.001	17.19567935	264.1	50.58413201	421.452
ddqn 1.0	0.001	16.96737049	264.1	36.9	351.827
mdqn2 0.0	0.001	14.07828206	212.08043	18.74729045	493.78
mdqn3 1.0	0.001	12.14972485	148.05634	8.572064894	357.792
mdqn3 0.0	0.001	9.328698486	179.06784	24.94178454	635.494
mdqn2 0.0	0.0001	9.311841471	202.07645	13.39635414	521.696
mdqn2 1.0	0.0001	5.351493407	127.04827	14.07459022	384.702
mdqn3 0.0	0.0001	4.406378303	102.03882	5.088549631	776.327
mdqn2 0.0	0.00001	1.603283236	61.02341	0.715069292	642.106
mdqn3 0.0	0.00001	0.872432773	12.00487	0.087176378	713.332
mdqn2 1.0	0.00001	0.692394861	12.00513	0.221167891	389.493

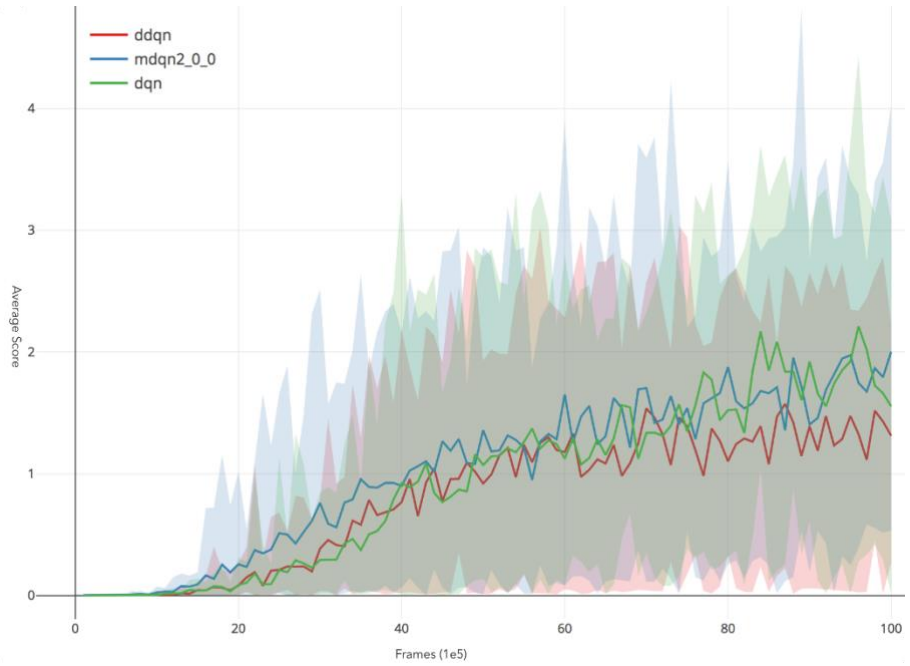

Figure 6. Comparison between DQN model types for Pong environment.

Table 5. Comparison of DQN, DDQN and MDQN models for 3D Raycast maze environment.

model type	lr	avg. score	var. score	time (min.)
mdqn2 0.0	0.00001	3.904359232	0.728045918	1213.991
dqn	0.00001	3.88654262	2.124993494	484.859
mdqn2 1.0	0.000001	3.7166532	0.154117942	533.389
ddqn	0.000001	3.713829593	1.524318234	524.65
ddqn	0.00001	3.638360789	1.662039807	521.975
mdqn2 0.0	0.00001	3.506246831	1.746571203	809.374
mdqn2 0.0	0.000001	3.345749731	2.749636472	978.638
ddqn	0.0001	3.267777864	2.889255991	523.012
mdqn2 1.0	0.0000001	3.247272282	0.576931468	500.424
mdqn2 1.0	0.00001	3.180342964	2.016163812	523.085
mdqn3 1.0	0.00001	3.056116361	2.339890159	872.317
dqn	0.000001	3.026868771	2.028895348	534.022
mdqn3 0.0	0.00001	2.807473511	2.395394139	1212.21
mdqn3 1.0	0.000001	2.770128326	0.714132328	864.442
mdqn3 0.0	0.000001	2.629530288	1.724929361	1152.146
mdqn2 0.0	0.0001	2.545370799	4.120312752	623.82
dqn	0.0001	2.24425396	2.153779645	516.078
mdqn3 1.0	0.0001	2.174641347	3.541216037	775.408
mdqn2 0.0	0.0001	2.157170755	3.235455294	899.93
mdqn2 1.0	0.0001	1.959047125	2.688871288	479.06
mdqn3 0.0	0.0001	1.678048035	3.272271359	1117.217
mdqn2 0.0	0.000001	1.452786487	1.887540531	809.63
mdqn2 1.0	0.00000001	1.399096696	1.827157866	495.813
mdqn2 0.0	0.0000001	0.178030303	0	463.67

4.3 Flappy Bird

Initially to test more than 28 hyper-parameters of DQN variants partial grid searches were done on combinations of parameters. Then benchmarking for one step changes were dome in each of hyper-parameters against initial parameters that are shown in Table 1.

Each set of parameters were repeated for at least 10 times to ensure repeatability as described in 4.2 section. By run, we mean full training of 10^7 frames with a defined set of hyper-parameters. RNN-ReLU were used as Q-value model in order to speed up training and compared DQN, DDQN and MDQN algorithms with full set of hyper-parameters as shown in Table 2, Table 3 and Table 4.

Original DQN outperformed DDQN and MDQN, but our version of MDQN slightly outperformed DDQN. This is nothing particularly surprising that DQN outperforms more advanced DDQN and MDQN because in previous studies it has also been shown that different algorithms excel in different environments. In some environments, DQN is more effective, but in others DDQN.

No significant improvements were found by applying some of more interesting architectures like Dueling Network or different activation functions in RNN like Leaky ReLU, ELU, and PreLU.

Regularization methods such as L1, L2, Dropout or Batch Normalization didn't improve performance. This could be the case because huge data set that is gathered from training environment in itself accomplishes normalization [4].

Because of the flexibility of open-source environments in PLE it was possible to produce Q-Value maps to track and compare the progress of different sets of hyper-parameters. An example of Q-Value maps is given in Figure 1.

4.4 Pong

For Pong and 3D raycast maze environments, in initial hyper-parameters optimizer was changed from "rmsprop" to "adam", because it gave better results without increasing processing time.

In case of Pong again DQN slightly outperformed MDQN and DDQN, but MDQN slightly outperformed DDQN as shown in Figure 6.

Q-Value maps were generated by manipulating the position of the ball in Pong environment on the frozen Q-Value model at checkpoints during training as shown in Figure 7.

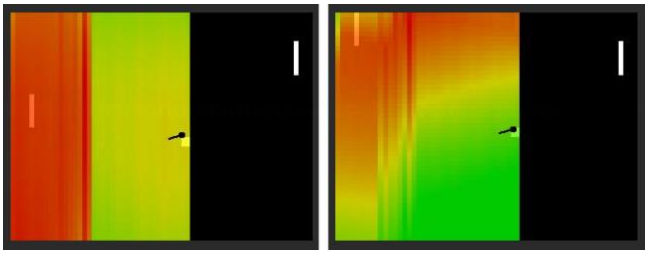

Figure 7. Pong Q-map before and after training. After training possible to see path of a ball trajectory.

4.5 3D Raycast maze

Finally, algorithms were benchmarked on "3D Raycast Maze" environment where instead of a low dimensional representation of a state, RGB 48x48 pixel input was used. In many environments, to save resources pixel grayscale representation would be recommended, but to make sure that exit door have a distinguishable difference in color form walls two channels were used per pixel red and green. The model consisted of ConvNet embedding and RNN layers.

All pixel inputs were normalized in a range $0.0 - 1.0$ instead of using byte value of $0 - 255$.

After the model has been trained GradCAM maps [20] were generated to visualize highest gradients in ConvNet as shown in Figure 8. These maps are more informative than Saliency Maps used in other Deep Reinforcement Learning papers [24]. These maps help us to understand what part of input pixel array is the most important for training. In this case, it was the exit door that gives the reward when reached.

Another way to reduce the dimensionality of the problem was to remove some of the actions available to an agent. Agent was allowed only to move ahead and make turns left and right, but not to go back and wait (do nothing).

Again Q-Value maps were constructed to visualize the progress of learning as shown in Figure 2. In this case, we manipulated a position of player around the maze and recorded Q-Values by rotating player's view around this position. Visual representation is a 2D top view for 3D maze.

Again, slight performance improvement were found using MDQN in a more complex environments like 3D Raycast maze as shown in Table 5.

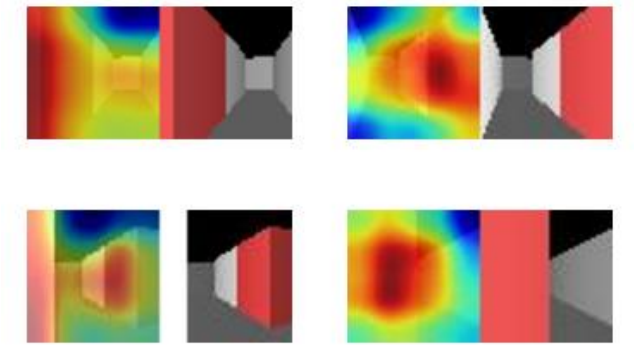

Figure 8. GradCAM maps of trained MDQN agent in 3D Raycast maze environment.
Images show attention on target in a 3D maze.

5. CONCLUSIONS

MDQN, a new Deep Reinforcement Learning algorithm was introduced that slightly outperforms DDQN in some environments. Still, in others, original DQN work better than both MDQN and DDQN.

Most of DQN variants that were tested have little or no significant effect on performance. New method to construct Q-Value maps were introduced by manipulating training environment. Q-Value maps are useful for assessing the progress of training. Results show that it is essential to run a sufficient number of repeated training runs for every set of parameters, because of the impact of random seed initialization and large variance in results.

6. APPENDIX

With this paper, spreadsheet is published of an average score in a game of Flappy Bird after 10^7 frames for each hyper-parameter with different learning rates. All hyper-parameters have been tested for DQN, DDQN and MDQN variants of algorithms. Results are available in public domain[4].

7. ACKNOWLEDGMENTS

Research has been completed with a support from High-Performance Computing Center of Riga Technical University.

8. REFERENCES

[1] Anschel, O., Baram, N. and Shimkin, N. 2016. Deep Reinforcement Learning with Averaged Target DQN. *NIPS Workshop*. (2016).

[2] Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J. and Kautz, J. 2017. GA3C: GPU-based A3C for Deep Reinforcement Learning. *ICLR*. (2017).

[3] Duan, Y., Chen, X., Houthooft, R., Schulman, J. and Abbeel, P. 2016. Benchmarking Deep Reinforcement Learning for Continuous Control. *ICML*. 48, (2016), 1329–1338.

[4] Geron, A. 2017. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

[5] Hasselt, H. van, Guez, A. and Silver, D. 2015. Deep Reinforcement Learning with Double Q-learning. *CoRR*. abs/1509.06461, (2015).

[6] Hasselt, H. van, Guez, A. and Silver, D. 2016. Deep Reinforcement Learning with Double Q-learning. *Proceedings of AAAI*. 13, (2016), 2094–2100.

[7] Hasselt, H.V. 2010. Double Q-learning. *Advances in Neural Information Processing Systems 23*. J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, eds. Curran Associates, Inc. 2613–2621.

[8] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D. and Meger, D. 2018. Deep Reinforcement Learning that Matters. (AAAI, 2018).

[9] Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M.G. and Silver, D. 2017. Rainbow: Combining Improvements in Deep Reinforcement Learning. *CoRR*. abs/1710.02298, (2017).

[10] Islam, R., Henderson, P., Gomrokchi, M. and Precup, D. 2017. Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *ICML*. (2017).

[11] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage and Anil Anthony Bharath 2017. A Brief Survey of Deep Reinforcement Learning. *IEEE Signal Processing Magazine*. (2017).

[12] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *US Patent 20170024643 A1*. (2015).

[13] Mnih, V. et al. 2015. Human-level control through deep reinforcement learning. *Nature*. 518, 7540 (2015), 529–533.

[14] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D. and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. *ICML*. 48, (2016), 1928–1937.

[15] Pfau, D. and Vinyals, O. 2016. Connecting Generative Adversarial Networks and Actor-Critic Methods. *NIPS Workshop on Adversarial Training*. (2016).

[16] Schaul, T., Quan, J., Antonoglou, I. and Silver, D. 2016. Prioritized Experience Replay. *ICLR*. (2016).

[17] Schulman, J., Levine, S., Moritz, P., Jordan, M.I. and Abbeel, P. 2015. Trust Region Policy Optimization. *ICML*. (2015), 1889–1897.

[18] Schulman, J., Moritz, P., Levine, S., Jordan, M.I. and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *ICLR*. (2016).

[19] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR*. abs/1707.06347, (2017).

[20] Selvaraju, R.R., Das, A., Vedantam, R., Cogswell, M., Parikh, D. and Batra, D. 2017. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *ICCV*. (2017).

[21] Tasfi, N. 2016. PyGame Learning Environment. *GitHub repository*. (2016).

[22] Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K. and de Freitas, N. 2017. Sample Efficient Actor-Critic with Experience Replay. *ICLR*. (2017).

[23] Wang, Z., Freitas, N. de and Lanctot, M. 2015. Dueling Network Architectures for Deep Reinforcement Learning. *CoRR*. abs/1511.06581, (2015).

[24] Wang, Z., Freitas, N. de and Lanctot, M. 2016. Dueling Network Architectures for Deep Reinforcement Learning. *ICML*. 16, (2016), 1995–2003.

[4] http://yellowrobot.xyz/full-survey-flappybird.pdf

DDQN parameters	lr=1e-05	lr=0.0001	lr=0.001
batch norm: false	1.533690858	0.000414183	0.000352965
batch norm: true	0.052704873	0.050553793	0.001360587
bellman gamma: 0.80	0.909917989	0.000442041	0.00033299
bellman gamma: 0.90	1.916720841	0.000477043	0.000387095
bellman gamma: 0.99	1.533690858	0.000414183	0.000352965
beta replay buffer: false	1.647591408	0.00041421	0.00043855
beta replay buffer: true	1.533690858	0.000414183	0.000352965
cumulative reward: false	0.610356388	0.10047675	0.0004505
cumulative reward: true	1.533690858	0.000414183	0.000352965
diff. states: false	1.533690858	0.000414183	0.000352965
diff. states: true	0.227873573	0.139044738	0.0186802
dropout: 0.0	1.533690858	0.000414183	0.000352965
dropout: 0.1	1.466420627	0.000413939	0.000354442
dueling arch.: false	1.533690858	0.000414183	0.000352965
dueling arch.: true	0.990586934	0.010944858	0.000481386
epsilon greedy: false	1.437370177	0.000377079	0.000323771
epsilon greedy: true	1.533690858	0.000414183	0.000352965
epsilon start,end: 1e-1, 1e-1	1.054991496	0.018243252	0.025396299
epsilon start,end: 1e-1, 1e-6	1.06638253	0.015038533	0.02246546
epsilon start,end: 1e-3, 1e-3	1.256475552	0.000369062	0.000344153
epsilon start,end: 1e-3, 1e-6	1.533690858	0.000414183	0.000352965
epsilon stuck: false	1.533690858	0.000414183	0.000352965
epsilon stuck: true	1.166306408	0.000483595	0.00074477
extra frame reward: 0.0	1.058717912	0	0
extra frame reward: 1e-5	1.533690858	0.000414183	0.000352965
frames back: 10	0.926641405	0.000392004	0.000364618
frames back: 5	1.533690858	0.000414183	0.000352965
frames before: 1e5	1.135056491	0.000397925	0.0003812
frames before: 5e4	1.533690858	0.000414183	0.000352965
grad clip.: 0.0	1.533690858	0.000414183	0.000352965
grad clip.: 1.0	1.486492849	7.633139692	12.45431387
grad clip.: 10.0	1.783358575	9.765907544	8.694875819
mini-batch: 16	1.352947647	0.000411727	0.00156524
mini-batch: 32	1.533690858	0.000414183	0.000352965
mini-batch: 64	1.056681304	0.000376604	0.000341958
mini-batch: 8	1.922057009	0.000311098	0.000389458
model: 1 states to n actions	1.533690858	0.000414183	0.000352965
model: n states to n act.	1.412973236	0.000294361	0.000401299
offline prebatch: false	1.533690858	0.000414183	0.000352965
offline prebatch: true	1.344636884	0.000348564	0.000363116
online: false	1.533690858	0.000414183	0.000352965
online: true	0.000716693	0.001188847	0.00207842
optimizer: adam	1.929185929	10.19414131	16.96737049
optimizer: rmsprop	1.533690858	0.000414183	0.000352965

DDQN paramaters	lr=1e-05	lr=0.0001	lr=0.001
priority replay buffer: proportional	0.838263139	0.000316778	0.000327183
priority replay buffer: ranked	1.533690858	0.000414183	0.000352965
reg.: l1 0.01	1.203800715	0.000285152	0.000381959
reg.: l2 0.01	0.986084538	0.000289221	0.000373825
reg.: none	1.533690858	0.000414183	0.000352965
replay buffer: 1e6	1.315573853	0.000322248	0.00035375
replay buffer: 5e5	1.533690858	0.000414183	0.000352965
rnn: elu	1.16391376	0.000382337	0.000334718
rnn: gru	1.046609285	42.97024986	0.398953106
rnn: leakyrelu	0.548797158	0.000282047	0.000397984
rnn: lstm	0.306796714	28.60916534	6.614351436
rnn: relu	1.533690858	0.000414183	0.000352965
rnn: selu	1.257988316	0.000375826	0.000345283
sarsa: false	1.533690858	0.000414183	0.000352965
sarsa: true	0.224702505	0.000387801	0.00032473
state prev. act., reward: false	1.533690858	0.000414183	0.000352965
state prev. act., reward: true	1.215485463	0.000606492	0.000375826
target network alpha: 1.0	1.533690858	0.000414183	0.000352965
terminal reward: -1e3	1.533690858	0.000414183	0.000352965
terminal reward: 0.0	0.000853557	0.000282642	0.000331196

MDQN3 paramaters	lr=1e-05	lr=0.0001	lr=0.001
batch norm: false	0.493159693	0.000411525	0.000302813
batch norm: true	0.18916225	0.023392921	0.008641977
bellman gamma: 0.80	0.667448615	0.000293686	0.000334672
bellman gamma: 0.90	0.657513296	0.000348975	0.00030927
bellman gamma: 0.99	0.493159693	0.000411525	0.000302813
beta replay buffer: false	0.501305723	0.000421983	0.000314868
beta replay buffer: true	0.493159693	0.000411525	0.000302813
cumulative reward: false	0.270724398	0.000377956	0.00031416
cumulative reward: true	0.493159693	0.000411525	0.000302813
diff. states: false	0.493159693	0.000411525	0.000302813
diff. states: true	0.31779793	0.213548615	0.039302415
dropout: 0.0	0.493159693	0.000411525	0.000302813
dropout: 0.1	0.602260763	0.000373934	0.000270495
dueling arch.: false	0.493159693	0.000411525	0.000302813
dueling arch.: true	0.504453143	0.000393069	0.000326828
epsilon greedy: false	0.630795214	0.000316404	0.000313741
epsilon greedy: true	0.493159693	0.000411525	0.000302813
epsilon start end: 1e-1 1e-1	0.587794046	0.017110157	0.024723262
epsilon start end: 1e-1 1e-6	0.665939117	0.016747024	0.022885654
epsilon start end: 1e-3 1e-3	0.704522112	0.000375783	0.0003061
epsilon start,end: 1e-3, 1e-6	0.493159693	0.000411525	0.000302813

MDQN3 paramaters	lr=1e-05	lr=0.0001	lr=0.001
epsilon stuck: false	0.493159693	0.000411525	0.000302813
epsilon stuck: true	0.878962391	0.000479067	0.000493813
extra frame reward: 0.0	0.42540338	0	0
extra frame reward: 1e-5	0.493159693	0.000411525	0.000302813
frames back: 10	0.119179613	0.00041888	0.000290899
frames back: 5	0.493159693	0.000411525	0.000302813
frames before: 1e5	0.461907203	0.000379282	0.000305497
frames before: 5e4	0.493159693	0.000411525	0.000302813
grad clip.: 0.0	0.493159693	0.000411525	0.000302813
grad clip.: 1.0	0.547549515	4.706386259	2.676778874
grad clip.: 10.0	0.86220963	5.774537436	6.148186995
mdqn: median	0.633033237	0.000350332	0.000302749
mdqn: min	0.740644674	0.000384621	0.0003037
mini-batch: 16	0.645145987	0.000378161	0.000336763
mini-batch: 32	0.493159693	0.000411525	0.000302813
mini-batch: 64	0.328581472	0.000438766	0.000276725
mini-batch: 8	0.719492781	0.000293855	0.000359393
model: 1 states to n actions	0.493159693	0.000411525	0.000302813
model: n states to n act.	0.881531711	0.000324639	0.000314943
offline prebatch: false	0.493159693	0.000411525	0.000302813
offline prebatch: true	0.561788219	0.000357654	0.000306432
online: false	0.493159693	0.000411525	0.000302813
online: true	0.000335295	0.002049978	0.000470529
optimizer: adam	0.659281593	5.541812022	12.14972485
optimizer: rmsprop	0.493159693	0.000411525	0.000302813
optimizer: sgd	0.000446	0.000452683	0.000421
pixels input: none	0.493159693	0.000411525	0.000302813
priority replay buffer: proportional	0.384257749	0.000327589	0.000333419
priority replay buffer: ranked	0.493159693	0.000411525	0.000302813
reg.: l1 0.01	0.542517945	0.000251858	0.00032751
reg.: l2 0.01	0.593226856	0.00053841	0.000285433
reg.: none	0.493159693	0.000411525	0.000302813
replay buffer: 1e6	0.624216444	0.000332877	0.000328961
replay buffer: 5e5	0.493159693	0.000411525	0.000302813
rnn: elu	0.375398893	0.000340552	0.000321809
rnn: gru	0.049088854	24.6588361	1.117617436
rnn: leakyrelu	0.085846548	0.000348879	0.000330276
rnn: lstm	0.051772933	16.17153479	6.362237775
rnn: relu	0.493159693	0.000411525	0.000302813
rnn: selu	0.510327967	0.000388453	0.000332236
sarsa: false	0.493159693	0.000411525	0.000302813
sarsa: true	0.371476081	0.000365584	0.000285508
state prev. act. reward: true	0.525153837	0.000486443	0.000268334
state prev. act., reward: false	0.493159693	0.000411525	0.000302813

MDQN3 paramaters	lr=1e-05	lr=0.0001	lr=0.001
target network alpha: 0.0	1.047481234	0.000454912	0.00048847
target network alpha: 0.5	0.3125444	0.009422778	0.000370117
target network alpha: 1.0	0.493159693	0.000411525	0.000302813
terminal reward: -1e3	0.493159693	0.000411525	0.000302813
terminal reward: 0.0	0.000789989	0.000302153	0.000279975

DQN paramaters	lr=1e-05	lr=0.0001	lr=0.001
batch norm: false	1.283649607	11.91111497	13.17070586
batch norm: true	0.056173539	0.045360783	0.010993424
bellman gamma: 0.80	1.703417127	6.185744115	1.528713372
bellman gamma: 0.90	2.137862124	19.36507549	11.39091096
bellman gamma: 0.99	1.283649607	11.91111497	13.17070586
beta replay buffer: false	1.703189311	10.04279246	12.89863154
beta replay buffer: true	1.283649607	11.91111497	13.17070586
cumulative reward: false	0.989741071	3.595820359	2.470743469
cumulative reward: true	1.283649607	11.91111497	13.17070586
diff. states: false	1.283649607	11.91111497	13.17070586
diff. states: true	0.134184431	0.167864504	0.134165564
dropout: 0.0	1.283649607	11.91111497	13.17070586
dropout: 0.1	1.330009148	0.000390686	0.000368321
dueling arch.: false	1.283649607	11.91111497	13.17070586
dueling arch.: true	0.027008964	0.001377553	0.000598506
epsilon greedy: false	2.153229242	10.52768314	15.05309331
epsilon greedy: true	1.283649607	11.91111497	13.17070586
epsilon start,end: 1e-1, 1e-6	1.131991281	5.228503262	4.929793323
epsilon start,end: 1e-3, 1e-6	1.283649607	11.91111497	13.17070586
epsilon stuck: false	1.283649607	11.91111497	13.17070586
epsilon stuck: true	1.24398021	3.328895906	3.526427801
extra frame reward: 0.0	1.1193924	12.62185465	11.46642777
extra frame reward: 1e-5	1.283649607	11.91111497	13.17070586
frames back: 10	1.565966324	12.67872818	6.766056991
frames back: 5	1.283649607	11.91111497	13.17070586
frames before: 1e5	1.314620253	11.72732532	8.484817536
frames before: 5e4	1.283649607	11.91111497	13.17070586
grad clip.: 0.0	1.283649607	11.91111497	13.17070586
grad clip.: 1.0	1.830272283	7.622029053	12.45487645
grad clip.: 10.0	1.510837858	12.12482097	8.745552219
mini-batch: 16	1.929140876	10.10765265	9.623069928
mini-batch: 32	1.283649607	11.91111497	13.17070586
mini-batch: 64	0.960969248	7.763781855	8.222028761
mini-batch: 8	1.556753631	8.285890256	6.141789924
model: 1 states to n actions	1.283649607	11.91111497	13.17070586
model: n states to n act.	1.266422601	0.000499987	0.000362667

DQN paramaters	lr=1e-05	lr=0.0001	lr=0.001
offline prebatch: false	1.283649607	11.91111497	13.17070586
offline prebatch: true	1.451969501	10.12259489	9.307812878
online: false	1.283649607	11.91111497	13.17070586
online: true	0.001364888	0.009441824	0.001442435
optimizer: adam	1.549136204	11.0958563	28.78679352
optimizer: rmsprop	1.283649607	11.91111497	13.17070586
optimizer: sgd	0.0005885	0.00060987	0.000464353
pixels input: none	1.283649607	11.91111497	13.17070586
pixels input: rgb			0.000497369
priority replay buffer: proportional	1.131824332	8.187889461	7.834152458
priority replay buffer: ranked	1.283649607	11.91111497	13.17070586
reg.: l1 0.01	1.469029549	0.000277902	0.000410069
reg.: l2 0.01	1.148608343	0.000322185	0.000314489
reg.: none	1.283649607	11.91111497	13.17070586
replay buffer: 1e6	1.208407003	11.2587372	9.307465784
replay buffer: 5e5	1.283649607	11.91111497	13.17070586
rnn: elu	1.42752302	12.02852927	9.555151711
rnn: gru	0.367137935	41.95876824	0.347982022
rnn: leakyrelu	0.46394823	0.000394567	0.000352919
rnn: lstm	0.432068508	38.19286308	6.333251634
rnn: relu	1.283649607	11.91111497	13.17070586
rnn: selu	1.549029797	0.000372131	0.000353642
sarsa: false	1.283649607	11.91111497	13.17070586
sarsa: true	0.394738604	0.144451096	0.129299982
state prev. act. reward: true	1.084134571	10.09573664	11.06724909
state prev. act., reward: false	1.283649607	11.91111497	13.17070586
target network alpha: 0.0	1.218063084	8.236031427	9.316638159
target network alpha: 0.5	1.465495324	7.837406845	12.07657609
target network alpha: 1.0	1.283649607	11.91111497	13.17070586
terminal reward: -1e3	1.283649607	11.91111497	13.17070586
terminal reward: 0.0	0.003241795	0.031652144	0.064477032

APPENDIX D - Paper 4

Exponential triplet loss

Evalds Urtans
Riga Technical University
Riga, Latvia
+37126401317
evalds.urtans@rtu.lv

Agris Nitkitenko
Riga Technical University
Riga, Latvia
agris.nikitenko@rtu.lv

Valters Vecins
Riga Technical University
Riga, Latvia
valters.vecins@rtu.lv

ABSTRACT

This paper introduces a novel variant of the Triplet Loss function that converges faster and gives better results. This function can separate class instances homogeneously through the whole embedding space. With Exponential Triplet Loss function we also introduce a novel type of embedding space regularization Unit-Range and Unit-Bounce that utilizes euclidean space more efficiently and resembles features of the cosine distance. We also examined factors for choosing the best embedding vector size for specific embedding spaces. Finally, we also demonstrate how new function can train models for one-shot learning and re-identification tasks.

CCS CONCEPTS

• **Theory of computation → Design and analysis of algorithms**;
• **Applied computing**;

KEYWORDS

Triplet loss, Feature embedding, Sample mining, One-shot learning, Identification, Re-identification

1 INTRODUCTION

Models that are capable of creating embedding representation that is somewhat disentangled and can be interpreted using distance metrics empowers many kinds of deep learning fields tasks starting with representation learning [1] [2], one-shot learning [3] [4], auto-encoders, generative models and reinforcement learning.

Nowadays, it is often not feasible to store all raw data from sensors and method for extracting and compressing only valuable data in the form of embedding is needed. Also, edge cases in data can be found using embedding queries thereby improving the quality of training data. Using the same queries these models can perform classification tasks on novel classes that have not been seen during training. Contrary to standard classification models, the precision of these models can be adjusted using cluster radiuses and other proximity metrics after they have been trained. These models also learn to generalize well and fewer samples to identify novel classes whereas classic classification models need a lot more data samples [5].

ICCDA 2020, March 9–12, 2020, Silicon Valley, CA, USA

ACM ISBN 978-1-4503-7644-0/20/03…$15.00
https://doi.org/10.1145/3388142.3388163

Sample-based methods like triplet loss [6], contrastive loss [7] [8] and N-tuple loss [8] has a goal to reduce the distance in embedding space between same class samples and increase the distance between different class samples. These methods require extensive sample mining methods and sample selection constraints to converge class representations into clusters. In this paper, we propose a new type of function to replace triplet loss that does not need sample selection constraints. This new function has an exponential shape to help converge faster samples that are further away from the desired state.

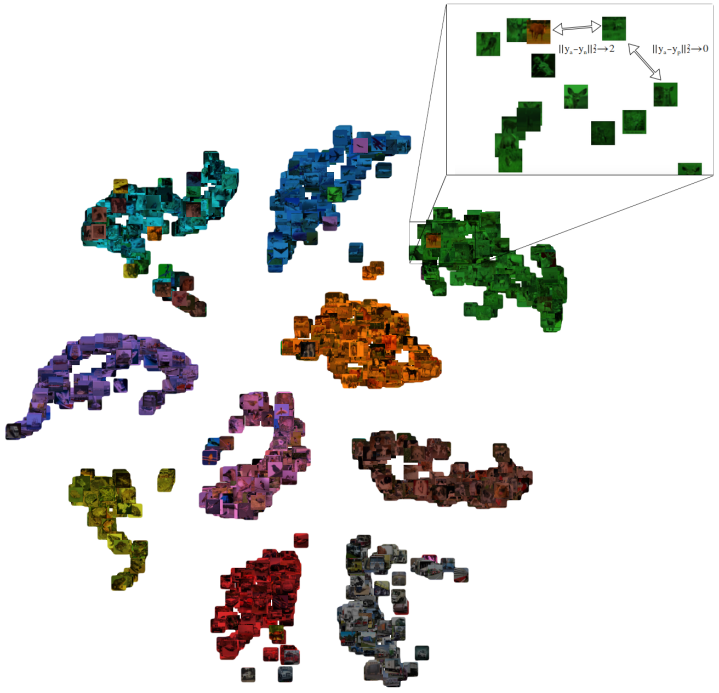

Figure 1. t-SNE embeddings of CIFAR10 trained with $\mathcal{L}_{exp}$. Colors denote different classes of samples. Exponential Triplet Loss decreases the distance between anchor y_a and positive same class sample y_p and increases the distance between different class sample y_n. Zoomed area shows sample from horse class that looks very similar to deer class samples.

2 RELATED WORK

Important research papers in Triplet Loss come from applications in face identification and re-identification tasks [6] [9].

Many variations of Triplet Loss have been explored by scientific community, notable mentions include Lifted Structured Loss [2], Histogram Loss [10], PDDM [11], n-pair loss [12], Triplet Ranking Loss [13], Additive Angular Margin Loss [14], Lossless Triplet Loss [15], N-Tuple Loss [8]. Similarly to our research, Margin Ranking Loss [16] worked on loss function shaping and described several shortcomings in standard Triplet Loss. One of the recent advances in Triplet Loss is Margin loss [17] that has some similar properties to our proposed loss function.

There has been a number of recent advances also in sample mining methods. Distance weighted sampling improves sample distributions in mini-batches [17], but is more useful in tandem with Margin Loss function. Doppelganger Mining [18] that takes into account most similar samples but with different classes. Hard example mining with auxiliary embeddings focuses on additional class features that can be used for better sampling strategy [19].

3 METHODOLOGY

Standard Triplet loss function (1) works with distances between embedding vectors y_p of same class as anchor embedding vector y_a and embedding vectors y_n of a different class than anchor embedding vector. Within the function, α is used as a margin between classes to not push them too far away and not to have them too close to each other.

$$\mathcal{L}_{std} = |\|y_a - y_p\|_2^2 - \|y_a - y_n\|_2^2 + \alpha|_+ \tag{1}$$

In Figure 2 of function $\mathcal{L}_{std}$ it is possible to see that many pairs in lower bound of loss function converge inconsistently.

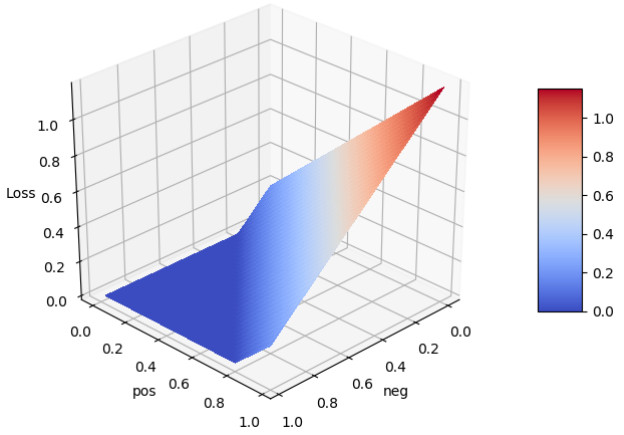

Figure 2. $\mathcal{L}_{std}$ function depending on positive pair $\|y_a - y_p\|_2^2$ (pos) and negative pair $\|y_a - y_n\|_2^2$ (neg) of embedding vectors

To have stable convergence of $\mathcal{L}_{std}$ so that negative embedding vector pairs are further away and positive are closer to each other at least one of two constraints must be enforced [9] [6] [20]:

(1) "Hard constraint" $\|y_a - y_p\|_2^2 + \alpha < \|y_a - y_n\|_2^2$

(2) "Semi-hard constraint" $\|y_a - y_n\|_2^2 < \|y_a - y_p\|_2^2$

After applying constraints in Figure 3 of function $\mathcal{L}_{std}$ it is possible to observe that area of sample combinations that guides model to converge is very small. To have convergence using $\mathcal{L}_{std}$ function it is necessary to have sample mining procedure before each mini-batch. Common approach is to use "batch hard" or "batch all" sample mining [21].

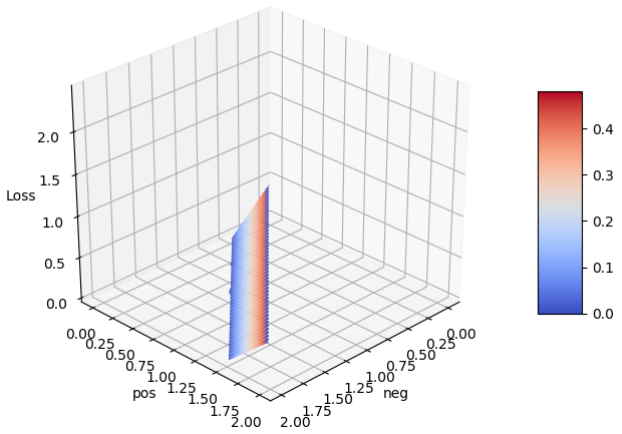

Figure 3. $\mathcal{L}_{std}$ function with constraints depending on positive pair $\|y_a - y_p\|_2^2$ (pos) and negative pair $\|y_a - y_n\|_2^2$ (neg) of embedding vectors

Our proposed Exponential Triplet Loss function $\mathcal{L}_{exp}$ has higher area where it can converge as seen in Figure 4. It also exploits exponential shape of function so that samples further away from desirable locations have much higher loss value. $\mathcal{L}_{exp}$ has asymmetric plateau where it does not draw closer pairs that are within half of embedding space maximum distance $\max(f_{emb}(x))$. As this loss function have a lot larger area of convergence it is less dependent on sample mining. Hyper-parameters C_{pos} and C_{neg} can be usually set to 1.0.

$$\mathcal{L}_{exp} = -C_{pos} \cdot log(1.0 - \frac{|emb_p - c_n|_+}{1 - c_n} + \epsilon)$$
$$-C_{neg} \cdot log(1.0 - \frac{|0.5 - emb_n|_+}{0.5} + \epsilon)$$

$$emb_p = \frac{y_p}{\max(f_{emb}(x))} \qquad emb_n = \frac{y_n}{\max(f_{emb}(x))} \tag{2}$$

It is not beneficial to push embedding vectors too far away from each other when the model would be used for the one-shot learning task. Sample of an unseen class that has not been introduced during training would "jump" between modalities of training data-set distribution. For one-shot learning task, homogeneous distribution of classes withing embedding space would be desirable. In order to enforce this, we propose overlap coefficient c_o that describes how much instances of different classes in data-set should overlap with

each other. Good value of c_o for clean data-sets is 1.5 that produces partial overlap in between all classes.

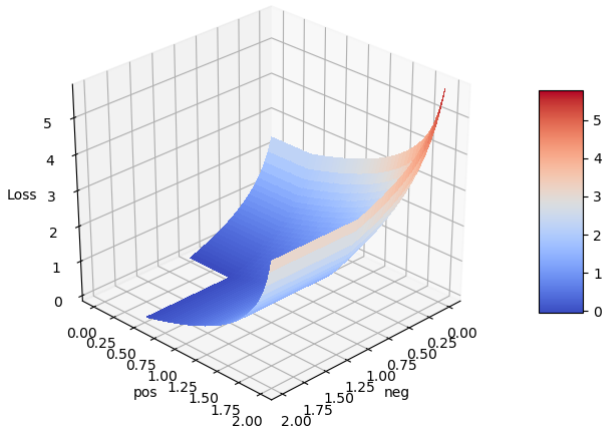

Figure 4. $\mathcal{L}_{exp}$ function depending on positive pair $\|y_a - y_p\|_2^2$ (pos) and negative pair $\|y_a - y_n\|_2^2$ (neg) of embedding vectors

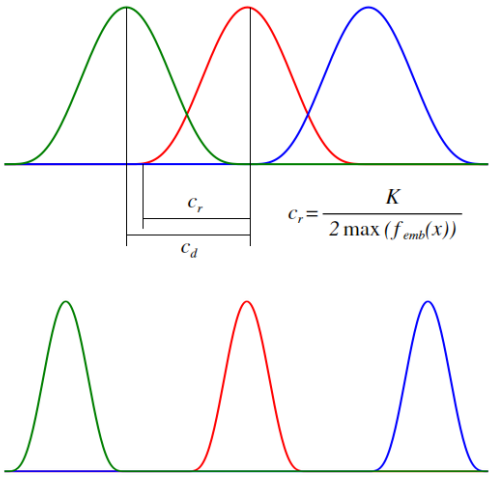

Figure 5. On the top example of overlapping class clusters in 1D with homogeneous embedding space. On the bottom example of non-homogeneous embedding space where new class samples will "jump" between clusters.

Function (3) evenly splits space into c_d embedding distances for each class in data-set. The number of classes in training data-set is K. When the model is used in the inference stage then the same K value is used as in the training stage even though the number of classes in inference might differ from training. $\mathcal{L}_{exp}$ can be used with cosine distance that has upper bound of $f_{emb} = 2$. It enables calculation of overlap distance that is calculated dividing this upper bound of distance by the number of classes and multiplied by overlap coefficient.

$$c_d = \frac{c_o \cdot K}{\max(f_{emb}(x))} \tag{3}$$

To use euclidean distances with $\mathcal{L}_{exp}$ it is necessary to have the upper bound of distance. It can be solved either by applying L2 normalization in which case it will always be $f_{emb} = 2 \cdot c_s$ depending on scale the maximum distance. But it constrains embedding vector positions within spherical space. We propose to use hybrid space that behaves as euclidean space when it is closer to the centre and when it reaches the radius of L2 then it behaves like spherical space. We call this function Unit-Range (4). Looking at embedding vectors using simple PCA it is possible to observe that Unit-Range space is more homogeneous than spherical space enforced by L2 normalization.

$$f_{emb}(x) = \begin{cases} c_s \frac{x}{|x|^2}, & \text{if } |x|^2 \geq 1 \\ x, & \text{otherwise} \end{cases} \tag{4}$$

Another function to normalize embedding space that is introduced by this paper is called Unit-Bounce (5). It resembles features of cosine distance as when embedding vector reaches the edge of a sphere it bounces back towards the centre of embedding space. When embedding reaches the opposite side of the sphere with a radius of c_s it bounces back towards centre again, similarly how the maximum angle between two vectors is always 180 degrees when calculating cosine distance.

$$f'_{emb}(x) = \begin{cases} f_{bounce}(x), & \text{if } |x|^2 \geq 1 \\ x, & \text{otherwise} \end{cases} \tag{5}$$

$$f_{bounce}(x) = \begin{cases} |x|^2 - \left\lfloor \frac{|x|^2}{c_s} \right\rfloor - c_s \frac{x}{|x|^2}, & \text{if } \left\lfloor \frac{|x|}{c_s} \right\rfloor \bmod 2 = 0 \\ c_s \frac{x}{|x|^2} - |x|^2 - \left\lfloor \frac{|x|^2}{c_s} \right\rfloor, & \text{otherwise} \end{cases} \tag{6}$$

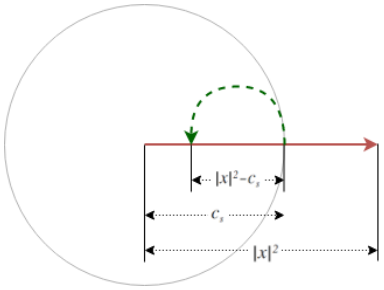

Figure 6. Illustration of Unit-Bounce embedding normalization function within L2 spherical space.

4 EXPERIMENTS

Proposed exponential triplet loss function $\mathcal{L}_{exp}$ has been tested on several image data-sets: MNIST, Fassion-MNIST, EMNIST, CIFAR10, CIFAR100 and VGGFace2 [22]. VGGFace2 dataset is typically used for face re-identification and verification tasks as well as one-shot learning evaluation. Implementation in PyTorch is available as an open-source code repository: https://github.com/evaldsurtans/exp-triplet-loss.

As an architecture of models we used pre-trained DenseNet-121 on ImageNet [23]. At the end of the model, we added the global average pooling function and max-out function with 16 linear units.

For all $\mathcal{L}_{exp}$ and $\mathcal{L}_{std}$ experiments grid search of best hyper-parameters were done to compare best results for each method. Experiments were repeated 5 times with the same set of hyper-parameters because in some cases convergence is sensitive to initialization.

For measuring accuracy two metrics were used:

- Closest accuracy (nearest neighbour) - after each epoch class cluster centres were calculated and each sample was assigned to the class closest to its embedding vector [24].
- Range accuracy - during each epoch class cluster centres were calculated and also their maximum distances within a class. Each sample added value of 1 to the class one-hot encoded vector so that embedding vector is in class cluster range. Afterwards, a one-hot encoded vector was L1 normalized to have probabilistic representation.

4.1 The dimensionality of a embedding vector

From empirical tests depending on an initialization method and the embedding space, we concluded that there is a dimension size limit for each combination these two parameters at which it does not improve performance of embedding models. It matches a simple experiment done by initializing different size vectors with a chosen pair of parameters shown in Figure 7. For example in L2 normalized spherical embedding space using cosine distances there are very small differences between vector size of 256 and 1024. For euclidean distances when L2 or Unit-Range even smaller embedding vector could be used at the size of in between 32 and 128. Experiments with training models with different embedding sizes confirmed this finding as shown in Figure 8. Intuitive explanation to this is that once distances between all samples at the beginning of training are same by random initialization then it is a good starting point for converging samples into homogeneous clusters covering the whole embedding space. At this point, it does not improve results to increase the dimension count of embedding vector. The same behaviour applies to cosine space as well as Euclidean space.

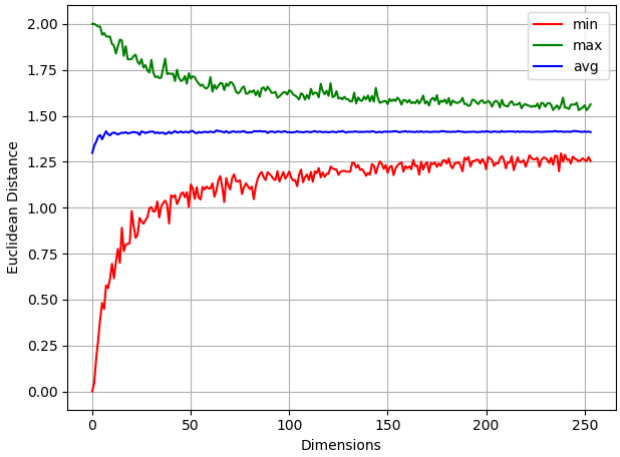

Figure 7. Cosine distances between 1000 sampled vectors that have been initialized by uniform distribution depending on dimension size.

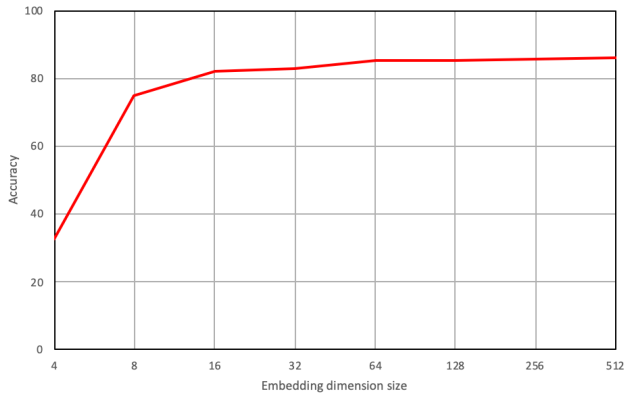

Figure 8. Accuracy of EMINST test data-set depending on dimension size of embedding vector. Unit-range and euclidean distances has been used in training.

4.2 Initialization of embedding vector

From empirical tests, we noticed that c_d had a higher effect on training if linear units of the embedding vector have been initialized using a uniform distribution. Such initialization yielded more uniform distribution of class clusters in the embedding space than more uni-modal initialization distributions. In Figure 9 distances between a sample and closest class cluster centre has been displayed. On left, there are distances before training, middle during training and on right after training. On top embedding, vectors are initialized using

Xavier initialization, but on the bottom using uniform initialization. Measurements have been taken from CIFAR10 data-set.

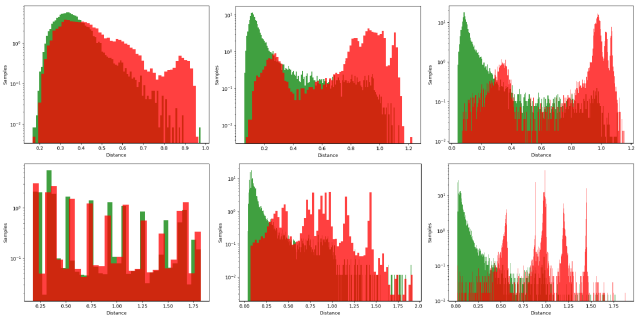

Figure 9. Comparison of xavier initialization on top and uniform initialization on bottom of embedding vectors. On left before training, in middle mid-part of training, on right after training.

4.3 Composite loss function

In order to improve performance of embedding models composite loss function were used. For all comparisons between $\mathcal{L}_{exp}$ and $\mathcal{L}_{std}$ composite loss functions were used. We introduced L2-constrained Softmax [25] with cross entropy $\mathcal{L}_{class}$ and center loss $\mathcal{L}_{center}$. We extended center loss (8) adding margin or radius of desired cluster to maintain (9). The idea is to discourage the collapse of embedding into one point in the embedding space. Within $\mathcal{L}_{class}$ input in Softmax function $f(x)$ is L2 normalized and scaled by s. Within $\mathcal{L}_{class}$ during training class instances are accumulated and c_{y_i} is calculated centre of cluster.

$$\mathcal{L}_{class} = -\sum_{i=1}^{M} y_i log \frac{e^{W_i^T s|f(x_i)|_2^2 + b_i}}{\sum_{j=1}^{C} e^{W_j^T s|f(x_i)|_2^2 + b_j}} \tag{7}$$

$$\mathcal{L}_{center'} = \sum_{i=1}^{M} ||x_i - c_{y_i}||_2^2 \tag{8}$$

$$\mathcal{L}_{center} = \sum_{i=1}^{M} |||x_i - c_{y_i}||_2^2 - \frac{c_d}{2}|_+ \tag{9}$$

$$\mathcal{L}_{\rfloor\updownarrow\surd} = \mathcal{L}_{exp} + C_{center}\mathcal{L}_{center} + C_{class}\mathcal{L}_{class} \tag{10}$$

4.4 One-shot learning

A model trained for one-shot learning task or re-identification task will work with novel classes in the test phase that has not been seen in the training phase. As $\mathcal{L}_{exp}$ models shown in Figure 10 and Figure 11 tend to have homogeneous embedding space they are able to cluster unseen class samples in between classes that have been seen during the training phase. Uniform initialization mentioned in the previous section is also beneficial for one-shot learning task where novel samples would be more evenly distributed in the embedding space.

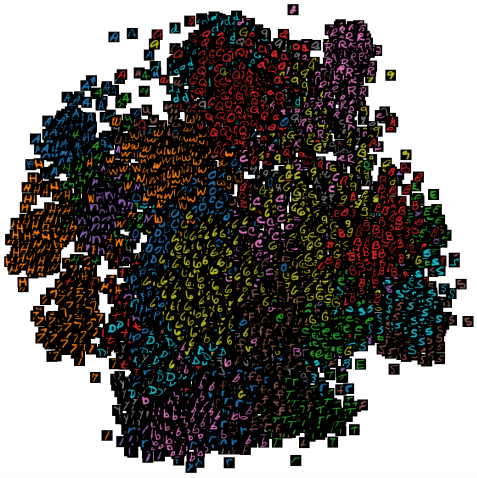

Figure 10. PCA of 2D of embeddings of class clusters seen in training. Model trained using $\mathcal{L}_{exp}$ and EMINST.

Figure 11. PCA of 2D of embeddings of class clusters not-seen in training. Model trained using $\mathcal{L}_{exp}$ and EMINST.

4.5 Results

Comparisons were made using the identification task wherein the inference phase there were the same classes as in the training phase. For each set of methods, hyper-parameters were tuned and multiple repeat training runs were made to compare only the best-performing model. All models were trained using state-of-the-art optimizer algorithm RAdam [26]. For some models, accuracy is close of those achieved by state of the art classification models. After analysis of mislabeled samples in the embedding space, it was possible to find many indistinguishable and wrong samples within training data-sets. All results of VGGFace2 have been obtained from re-identification task and one-shot learning task where test data-set classes were not included in train data-set. For VGGFace2 we used only 1000 samples of each class in training and testing data-sets with lower resolution of 128x128 pixels.

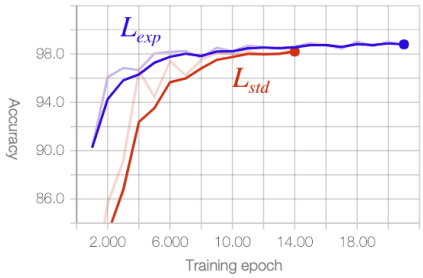

Figure 12. Comparison of convergence speed between $\mathcal{L}_{exp}$ and $\mathcal{L}_{std}$ on Fassion-MINST data-set.

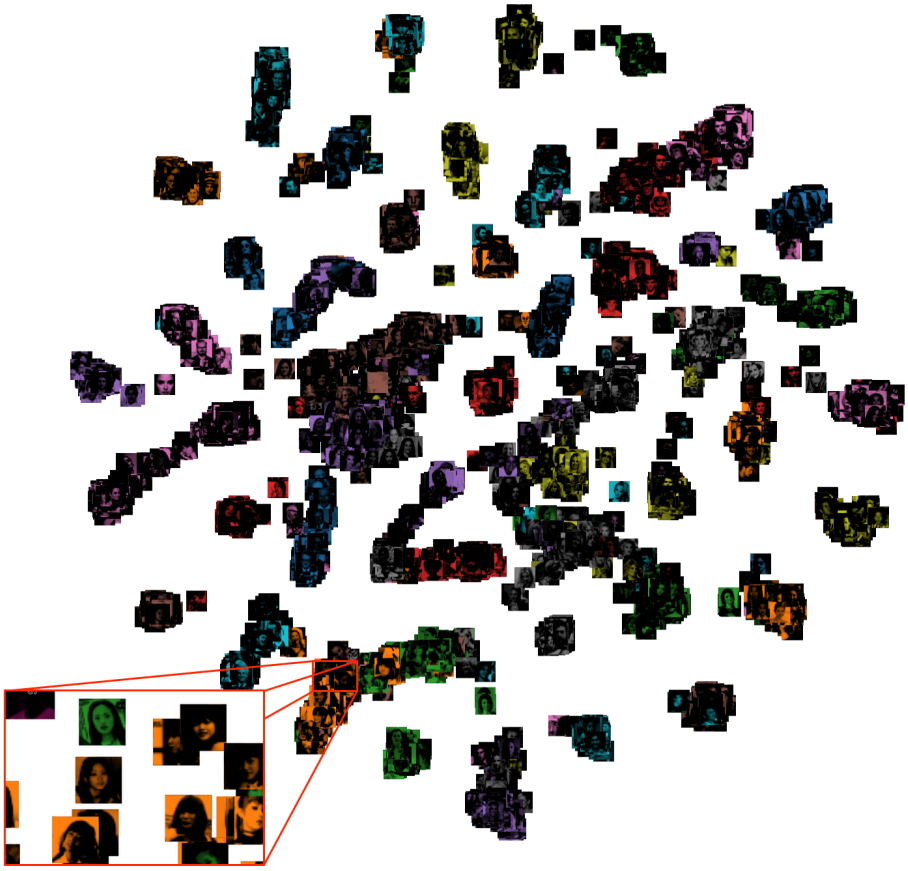

Figure 13. t-SNE of 50 color-coded samples of VGGFace2 test data-set trained by $\mathcal{L}_{exp}$. In lower-left corner sample given where model clusters together close samples of similar Asian woman face with dark hair.

Table 1. Comparison of accuracy between models trained with different types of loss functions. Accuracy calculated by closest cluster centre to a sample. All models use best fitted hyper-parameters and superior Unit-Range type of embedding normalization.

Loss func. / Accuracy %	MNIST	Fassion-MNIST	EMINST	CIFAR10	VGGFace2
L_{std}	**99.6**	91.4	82.0	56.2	77.4
$L_{std} + L_{class}$	**99.6**	92.1	85.0	79.8	76.3
$L_{std} + L_{center}$	97.5	71.5	61.7	52.1	76.4
$L_{std} + L_{center} + L_{class}$	97.7	82.0	70.9	62.8	78.6
L_{exp}	**99.6**	92.7	82.7	85.7	85.0
$L_{exp} + L_{class}$	**99.6**	**93.1**	85.2	87.2	84.1
$L_{exp} + L_{center}$	**99.6**	**93.1**	85.7	85.3	84.0
$L_{exp} + L_{center} + L_{class}$	**99.6**	**93.1**	**86.0**	**87.3**	**85.7**

5 CONCLUSIONS

Proposed Exponential Triplet Loss function provides an easier way to train embedding models. With this function, models converge faster and have higher accuracy and class separation. They produce embeddings in better utilized and more homogeneous Unit-Range and Unit-Bounce embedding spaces than in L2 spherical embedding space. The embedding normalization function Unit-Bounce resembled the same properties as cosine distances but using euclidean distances. Also, the training relies on less sample mining as the convergence space covers more of the sample space. The embedding models in this paper generalizes well also in one-shot learning task where novel class samples grouped in clusters even though they were not seen during the training phase.

6 ACKNOWLEDGMENTS

Research has been completed with support from High-Performance Computing Center of Riga Technical University that provided 12 nVidia K40 GPUs and 8 nVidia V100 GPUs. Special thanks to Lauris Cikovskis.

REFERENCES

[1] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, , T. M. Hospedales, "Learning to Compare: Relation Network for Few-Shot Learning," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 1199–1208. http://openaccess.thecvf.com/content_cvpr_2018/html/Sung_Learning_to_Compare_CVPR_2018_paper.html

[2] H. O. Song, Y. Xiang, S. Jegelka, , S. Savarese, "Deep Metric Learning via Lifted Structured Feature Embedding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4004–4012. https://doi.org/10.1109/CVPR.2016.434

[3] M. Bucher, S. Herbin, , F. Jurie, "Improving Semantic Embedding Consistency by Metric Learning for Zero-Shot Classiffication," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V*, pp. 730–746. https://doi.org/10.1007/978-3-319-46454-1_44

[4] Y. Yuan, K. Yang, , C. Zhang, "Hard-Aware Deeply Cascaded Embedding," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 814–823. https://doi.org/10.1109/ICCV.2017.94

[5] C. Wang, X. Lan, , X. Zhang, "How to train triplet networks with 100K identities?" pp. 1907–1915.

[6] F. Schroff, D. Kalenichenko, , J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 815–823. https://doi.org/10.1109/CVPR.2015.7298682

[7] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, , R. Shah, "Signature Verification Using A "Siamese" Time Delay Neural Network," vol. 7, no. 4, pp. 669–688. https://doi.org/10.1142/S0218001493000339

[8] S. Chopra, R. Hadsell, , Y. LeCun, "Learning a Similarity Metric Discriminatively, with Application to Face Verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pp. 539–546. https://doi.org/10.1109/CVPR.2005.202

[9] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, , Y. Wu, "Learning Fine-Grained Image Similarity with Deep Ranking," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 1386–1393. https://doi.org/10.1109/CVPR.2014.180

[10] E. Ustinova , V. S. Lempitsky, "Learning Deep Embeddings with Histogram Loss," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 4170–4178. http://papers.nips.cc/paper/6464-learning-deep-embeddings-with-histogram-loss

[11] C. Huang, C. C. Loy, , X. Tang, "Local Similarity-Aware Deep Feature Embedding," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1262–1270. http://papers.nips.cc/paper/6368-local-similarity-aware-deep-feature-embedding

[12] B. Amos, B. Ludwiczuk, , M. Satyanarayanan, "OpenFace: A general-purpose face recognition library with mobile applications," in *CMU*.

[13] M. Ye , Y. Guo, "Deep Triplet Ranking Networks for One-Shot Recognition," vol. abs/1804.07275. http://arxiv.org/abs/1804.07275

[14] J. Deng, J. Guo, , S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," vol. abs/1801.07698. http://arxiv.org/abs/1801.07698

[15] Marc-Olivier Arsenault. Lossless Triplet loss. https://towardsdatascience.com/lossless-triplet-loss-7e932f990b24

[16] D. P. Vassileios Balntas, Edgar Riba , K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proceedings of the British Machine Vision Conference (BMVC)*, E. R. H. Richard C. Wilson , W. A. P. Smith, Eds. BMVA Press, pp. 119.1–119.11. https://dx.doi.org/10.5244/C.30.119

[17] R. Manmatha, C.-Y. Wu, A. J. Smola, , P. Krähenbühl, "Sampling Matters in Deep Embedding Learning," pp. 2859–2867.

[18] E. Smirnov, A. Melnikov, S. Novoselov, E. Luckyanets, , G. Lavrentyeva, "Doppelganger Mining for Face Representation Learning," in *2017 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017, Venice, Italy, October 22-29, 2017*, pp. 1916–1923. https://doi.org/10.1109/ICCVW.2017.226

[19] E. Smirnov, A. Melnikov, A. Oleinik, E. Ivanova, I. Kalinovskiy, , E. Luckyanets, "Hard Example Mining With Auxiliary Embeddings," in *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 37–46. http://openaccess.thecvf.com/content_cvpr_2018_workshops/w1/html/Smirnov_Hard_Example_Mining_CVPR_2018_paper.html

[20] O. M. Parkhi, A. Vedaldi, , A. Zisserman, "Deep Face Recognition," in *BMVC*.

[21] A. Hermans, L. Beyer, , B. Leibe, "In defense of the triplet loss for person re-identification," vol. abs/1703.07737.

[22] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, , A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," pp. 67–74.

[23] G. Huang, Z. Liu, , K. Q. Weinberger, "Densely Connected Convolutional Networks," pp. 2261–2269.

[24] K. Q. Weinberger , L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," vol. 10, pp. 207–244. https://dl.acm.org/citation.cfm?id=1577078

[25] R. Ranjan, C. D. Castillo, , R. Chellappa, "L2-constrained Softmax Loss for Discriminative Face Verification," vol. abs/1703.09507. http://arxiv.org/abs/1703.09507

[26] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, , J. Han, "On the variance of the adaptive learning rate and beyond," vol. abs/1908.03265.

APPENDIX E - Paper 5

asya: Mindful verbal communication using deep learning

Pre-print Version, August 20, 2020

Evalds Urtans
Riga Technical University
Riga, Latvia
evalds@asya.ai

Ariel Tabaks
University of Central Lancashire
Preston, Lancashire, UK
ariel@asya.ai

ABSTRACT

asya is a mobile application that consists of deep learning models which analyze spectra of a human voice and do noise detection, speaker diarization, gender detection, tempo estimation, and classification of emotions using only voice. All models are language agnostic and capable of running in real-time. Our speaker diarization models have accuracy over 95% on the test data set. These models can be applied for a variety of areas like customer service improvement, sales effective conversations, psychology and couples therapy.

CCS CONCEPTS

• **Theory of computation → Design and analysis of algorithms**;
• **Applied computing**;

KEYWORDS

Deep Learning, Triplet loss, ConvNet, ResNet, DenseNet, Mel-Spectra, Speaker diarization, Emotion detection, NLP

1 INTRODUCTION

asya is a mobile application that listens to a person's voice and provides private feedback on a person's verbal communication. It gives metrics on how much a person listens and speaks, how long are a person's sentences and utterances, how fast a person speaks, how positive is the tone of a person, the confidence level of a person's voice based on the tone and other metrics. It is deployed as a web service and stand-alone application. These models are being applied to a variety of tasks starting from customer service evaluation to analysis of private conversations and couple's relationships therapy.

Neuroscientists Andrew Newberg, M.D., and Mark Waldman, have identified through brain scans and from other studies that if everyday verbal interactions is coupled with increased moment-to-moment awareness the results can lead to increased levels of trust building, resolved conflicts, increased intimacy and other benefits [3]. The findings show that people can benefit from speaking less, shorter, and slower as human brain short-term memory holds only about four "chunks" of information, which translates to speaking time under 30 seconds [3]. Furthermore, when people practice the 30 second rule, they can train themselves to increase awareness of filtering out lower quality information. In addition, when speaking is kept with brevity in mind, the emotional centers of the brain that can be triggered by certain words are less likely to lead the speaker to sabotage the conversation. The research also shows that exercising awareness not only helps to increase connection with other people, but also suppresses the brain's ability to generate feelings of anxiety, irritability or stress.

The tone of voice, emotions, and the way how a person speaks are as important as the content that a person speaks. For example, one could imagine how one could say the phrase "you are such a fool" in a way that could offend someone or in a way that could be even playful and funny.

2 RELATED WORK

In the last decade, there has been great progress in natural language processing (NLP) models to do the tasks like "text-to-speech", "speech-to-text", translation, semantic and syntactic understanding of language [21] [23] [1] [9]. However, only in recent years there has been emerging research to use deep learning models to analyze human's voice biometric features that are linked to psychology [12] [5]. Before it has been done using classical machine learning models that yielded in lesser results [13]. Voice features that typically are used for analysis are Mel-frequency filter banks, Log Mel spectrograms, Mel-frequency cepstrum (MFCC), or even raw waveforms in combination with audio envelopes. Mel-frequency filter banks are filters that are applied to spectrum calculated by Fast Fourier Transform (FFT) to simulate specific amplitudes of sounds at different frequencies that are audible to the human ear. The audio spectrum is much broader than that what human ear can perceive, but other frequencies of sound are less likely to contain useful information for NLP tasks.

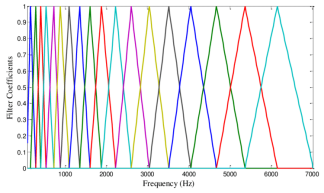

Figure 1: Mel-frequencies filter bank to emulate human's ear perception.

Recently, also some progress has been done into recognizing emotions form a person's voice. Historically, there have been very little datasets available for voice alone, but recent techniques using transfer learning enabled to accumulate considerable datasets with unsupervised learning to analyze emotions in a human's voice [2].

https://asya.ai, August 20, 2020, Riga, Latvia

Some of the recent works have used deep Convolutional Networks (ConvNet) [10] to extract features from human's voice spectra and classify 8 basic emotions: Happiness, Sadness, Anger, Fear, Disgust, Surprise, Boredom and Neutrality. Even with the basic ConvNet model, it has been possible to surpass human reference accuracy on detecting emotions in a person's voice. For example, humans on average were able to detect happiness in voice with 65% precision, whereas ConvNet model was able to detect it with 100% precision [19], [15].

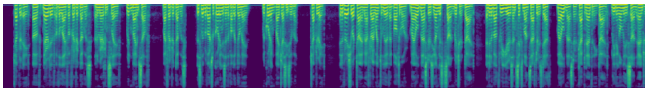

Figure 2: Mel-spectogram of a person speaking (processed input of asya models).

3 METHODOLOGY

asya uses deep learning models that take as an input
Mel spectrograms and other features from raw voice recordings. Deep Residiual Networks (ResNet) [7] and DenseNet [8] models have been applied as feature encoders. ConvNet models are deep artificial neural network models that have very similar results when experimentally compared to human retina natural neural networks [11]. At first layers of the model, they extract basic features like Gabor patches and edges, but the deeper they go they extract more general features. For example, for face detection task, first they would detect features like nose and eyes, but then in deeper layers faces as a whole. These models are very deep with usually more than 32 layers. ConvNet models with residual connections (ResNet) allow error to flow freely using back-propagation algorithm without vanishing gradient problem. In case of DenseNet, there are even more connections and better flow of gradient of error through the model. asya models have been trained on multiple large private datasets from different speakers and languages using DenseNet models and other proprietary models.

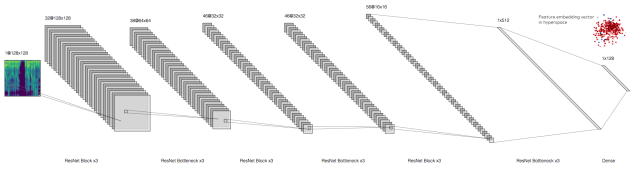

Figure 3: Schematic illustration of ConvNet model for speaker feature embedding using Mel-spectogram sample.

3.1 Speaker diarization

Speaker diarization (identification of different speakers in parts of utterances) has been done primary using i-vector, d-vector [17], x-vector [18] based models.

More recently, RNN based models also have been applied like UIS-RNN [22].

It has also been done using triplet loss or contrastive loss and cosine similarity of embedding vectors (fingerprint vectors of human voice) [12] [5] [16]. asya uses Exponential Triplet Loss function and clustering of speaker embedding to achieve speaker diarization and speaker re-identification in one step [20].

asya models are utterance (phrase) and language independent, whereas, for example, Google Home recognize speaker by specific phrase like "Ok, Google". It means that asya models are capable of identifying a person's voice at any point in natural conversations. To improve training results of speaker diariaztion models data have been split into multiple parts. asya has been trained as a set of hierarchical models that first predict if the audio in a given window is a noise or speech, then if it is a man or woman and finally does feature embedding of person's voice. During the testing, we use the center of the mass of a person's voice embedding vector to estimate a probability of voice sample belonging to a particular person.

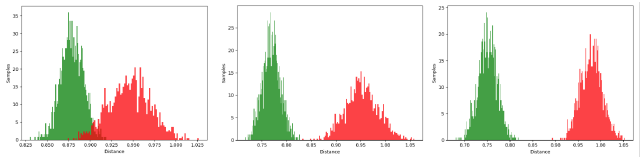

Figure 4: Histograms of cosine distance between speaker embedding vectors acquired by asya male models. Left in the beginning of training, right after training have been completed. Green are distances between samples form same speaker. Red are distances between samples form different speakers.

Figure 5: Representation of speaker embedding vectors from asya models in 3D space (Spherical PCA). Colors denote different speaker samples in test data set.

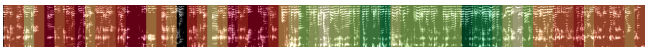

Figure 6: Segmentation of audio sample for speaker diarization task for 2 male speakers. Red frames are furthest away from speaker feature embedding vector and green frames are closest thus identifying target speaker [1].

3.2 Emotion classification

Speaker emotion classification task is also done using the same deep learning feature extractor as for speaker diarization task, but with softmax loss function at the end [19].

Standard academic datasets of classified emotional states of audio are read to validate approaches. For example, traditional German EmoDB dataset [6] contains only 500 samples of 10 speakers.

The methodology for our work initially has been based on EmoVoxCeleb [2]. EmoVoxCeleb is trained on FERPlus [4] dataset of still pictures of human FER (Facial Expressions) in different emotional states and then applied to a larger VoxCeleb dataset of videos [14]. These emotional states are classified as Paul Ekman's 8 basic emotions.

Even though we can achieve state-of-art results in academic data-sets we had to create our own proprietary methodology and dataset to to reach similar performance in production systems.

To acquire more training, data transfer learning and unsupervised learning have been used to scrape public data from video sites in the internet.

Table 1: Preliminary results of emotion classification using asya models

emotion	Human [14]	EmoVoxCeleb	asya
happiness	84	35	62
sadness	81	71	75
anger	97	72	75
fear	84	35	75
surprise	81	36	69
disgust	67	67	75
neutral	93	N/A	69

4 RESULTS

Asya models are currently in development and are being tested using mobile application in natural conversations to improve couples relationships through conversations in a similar manner, how does couple therapy would work.

Set of hierarchical models are executed in real-time in less than 500 ms. for every 1 sec. frame on consumer grade GPU server. It is also possible also to deploy and execute these models on flagship mobile phones with machine learning specialized processing units.

Table 2: Preliminary results of speaker re-identification and other voice classification tasks of asya models

model	accuracy
noise detection	99.2
gender detection	89.3
male speaker re-identification	87.4
female speaker re-identification	88.4

5 CONCLUSIONS

The proposed models are capable of analyzing human's voice in real-time. asya is able to detect noise in audio samples and process only parts with a human's voice. asya models are able to detect a human's perception of the gender of the speaker with high precision. Finally, it is able to produce a unique embedding vector for each person's voice to combine speaker diarization and reidentification tasks in a single step. Furthermore, asya models are able to detect Ekman's basic human emotions from language and utterance independent data.

There can be a wide range of use cases for asya models. It has been successfully deployed to improve communication skills and encourage mindful communication in a commercial product https://asya.ai. Asya is being developed also to improve public speaking skills and provide feedback for psychologists about their sessions with patients. Asya models also can be used to monitor customer experience in customer service-centered businesses like phone hotlines, post offices, stores, telemarketing, etc. Finally, they could be used to identify persons of interest in large databases of audio recordings, but there are even more use cases than listed in this paper.

REFERENCES

[1] Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. [n. d.]. Weighted Transformer Network for Machine Translation. abs/1711.02132 ([n. d.]). arXiv:1711.02132 http://arxiv.org/abs/1711.02132

[2] Samuel Albanie, Arsha Nagrani, Andrea Vedaldi, and Andrew Zisserman. [n. d.]. Emotion Recognition in Speech Using Cross-Modal Transfer in the Wild. In *2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018* (2018). 292–301. https://doi.org/10.1145/3240508.3240578

[3] Andrew Newberg M.D and Mark Robert Waldman. [n. d.]. *Words Can Change Your Brain: 12 Conversation Strategies to Build Trust, Resolve Conflict, and Increase Intimacy.*

[4] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. [n. d.]. Training Deep Networks for Facial Expression Recognition with Crowd-Sourced Label Distribution. In *ACM International Conference on Multimodal Interaction (ICMI)* (2016).

[5] Hervé Bredin. [n. d.]. TristouNet: Triplet Loss for Speaker Turn Embedding. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017* (2017). 5430–5434. https://doi.org/10.1109/ICASSP.2017.7953194

[6] Felix Burkhardt, Astrid Paeschke, Melissa A Rolfes, Walter F. Sendlmeier, and Benjamin Weiss. [n. d.]. A Database of German Emotional Speech. In *INTERSPEECH* (2005).

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. [n. d.]. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016* (2016). 770–778. https://doi.org/10.1109/CVPR.2016.90

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. [n. d.]. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* (2017). 2261–2269. https://doi.org/10.1109/CVPR.2017.243

[9] Takuhiro Kaneko and Hirokazu Kameoka. [n. d.]. Parallel-Data-Free Voice Conversion Using Cycle-Consistent Adversarial Networks. abs/1711.11293 ([n. d.]). arXiv:1711.11293 http://arxiv.org/abs/1711.11293

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. [n. d.]. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[11] Jonas Kubilius, Stefania Bracci, and Hans P. Op de Beeck. [n. d.]. Deep Neural Networks as a Computational Model for Human Shape Sensitivity. 12, 4 ([n. d.]). https://doi.org/10.1371/journal.pcbi.1004896

[12] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu. [n. d.]. Deep Speaker: An End-to-End Neural Speaker Embedding System. abs/1705.02304 ([n. d.]). arXiv:1705.02304 http://arxiv.org/abs/1705.02304

[13] Benoˆ[U+FFFD]t Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gaël Richard. [n. d.]. YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010* (2010). 441–446. http://ismir2010.ismir.net/proceedings/ismir2010-75.pdf

[14] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. [n. d.]. VoxCeleb: A Large-Scale Speaker Identification Dataset. In *INTERSPEECH* (2017).

[15] Yafeng Niu, Dongsheng Zou, Yadong Niu, Zhongshi He, and Hua Tan. [n. d.]. A Breakthrough in Speech Emotion Recognition Using Deep Retinal Convolution Neural Networks. abs/1707.09917 ([n. d.]). arXiv:1707.09917 http://arxiv.org/abs/1707.09917

[16] Florian Schroff, Dmitry Kalenichenko, and James Philbin. [n. d.]. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015* (2015). 815–823. https://doi.org/10.1109/CVPR.2015.7298682

[17] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur. [n. d.]. Deep Neural Network Embeddings for Text-Independent Speaker Verification. In *INTERSPEECH* (2017).

[18] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. [n. d.]. X-Vectors: Robust DNN Embeddings for Speaker Recognition. ([n. d.]), 5329–5333.

[19] Somayeh Shahsavarani. [n. d.]. Speech Emotion Recognition Using Convolutional Neural Networks.

[20] Evalds Urtans, Agris Nikitenko, and Valters Vecins. [n. d.]. Exponential Triplet Loss. In *ICCDA* (2020).

[21] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. [n. d.]. WaveNet: A Generative Model for Raw Audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016* (2016). 125. http://www.isca-speech.org/archive/SSW_2016/abstracts/ssw9_DS-4_van_den_Oord.html

[22] Aonan Zhang, Quan Wang, Zhenyao Zhu, John W. Paisley, and Chong Wang. [n. d.]. Fully Supervised Speaker Diarization. ([n. d.]), 6301–6305.

[23] Cong Zhou, Michael Horgan, Vivek Kumar, Cristina Vasco, and Dan Darcy. [n. d.]. Voice Conversion with Conditional SampleRNN. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*. (2018). 1973–1977. https://doi.org/10.21437/Interspeech.2018-1121