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

Summary of the Doctoral Thesis

Scientific supervisor
professor Dr. sc. ing.
Agris Nikitenko

RTU Press
Riga 2021

Urns . Function shaping in deep learning. Promocijas darba kopsavilkums. Riga: RTU Press, 2021. 73 p.

Published in accordance with the decision of the Promotion Council RTU P-xxx of December 1, 2021, No. xx.

ISBN xxx-xxxx-xx-xxx-x (pdf)

DOCTORAL THESIS PROPOSED TO RIGA TECHNICAL UNIVERSITY FOR THE PROMOTION TO THE SCIENTIFIC DEGREE OF DOCTOR OF SCIENCE

To be granted the scientific degree of Doctor of Science (Ph. D.), the present Doctoral Thesis has been submitted for the defence at the open meeting of RTU Promotion Council on December 1, 2021 at the Faculty of Computer Science and Information Technology of Riga Technical University, remotely.

OFFICIAL REVIEWERS

Dr. William Sayers,
University of Gloucestershire,
UK;

Dr. Houxiang Zhang,
Norwegian University of Science and Technology,
Norway

DECLARATION OF ACADEMIC INTEGRITY

I hereby declare that the Doctoral Thesis submitted for the review to Riga Technical University for the promotion to the scientific degree of Doctor of Science (Pf. D) is my own. I confirm that this Doctoral Thesis had not been submitted to any other university for the promotion to a scientific degree.

Name Surname .... (signature)
Date: ....

The Doctoral Thesis has been written in Latvian/English. It consists of an Introduction; 8 Chapters; 27 figures; 14 tables; x appendices; the total number of pages is 165, including appendices. The Bibliography contains 136 titles.

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 analytically derived from 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 in the face re-identification 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 re-identification, 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 Summary of the Doctoral Thesis 14

2 IMPROVING THE PERFORMANCE OF FUNCTIONS USING DEEP LEARNING MODELS **17**
- 2.1 Value Iteration Algorithm 17
- 2.2 ConvNet and UNet Models 20
- 2.3 RNN Models 24
- 2.4 UNet-RNN-Skip Model 27

3 FUNCTION SHAPING IN DEEP REINFORCEMENT LEARNING **31**
- 3.1 Q-Value and Policy Gradient Functions for Reinforcement Learning 31
- 3.2 Deep Q-Network model and Loss Function 32
- 3.3 Multi Deep Q-Network Model and Loss Function 35

4 FUNCTION SHAPING IN DEEP METRIC LEARNING **38**
- 4.1 Zero-Shot Learning and Re-identification Task 38
- 4.2 Triplet Loss Function 40
- 4.3 Exponential Triplet Loss Function 42

5 EXPERIMENTAL RESULTS AND APPLICATIONS **45**
- 5.1 Results of UNet-RNN-Skip Model 45
- 5.2 Results of Multi Deep Q-Network Loss Function 49
- 5.3 Results of Exponential Triplet Loss Function 55
- 5.4 Practical Applications 57

6 FUTURE RESEARCH **59**

7 CONCLUSIONS **60**

ACKNOWLEDGEMENTS **61**

BIBLOGRAPHY **62**

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 have been evaluated on computer game environments to achieve higher score. Similarly, loss functions in Deep Metric Learning have 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 Equation (1) is true and when 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.

$$|\frac{\mathcal{L}(f_\theta(x), y)_i}{\mathcal{L}(f_\theta(x), y)_{i-1}}| < \delta \tag{1}$$

The objectives of the Thesis are as follows:

1. Perform a 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 are as follows:

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 outperforms 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 Subsection **??**.

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 Fig. 1.

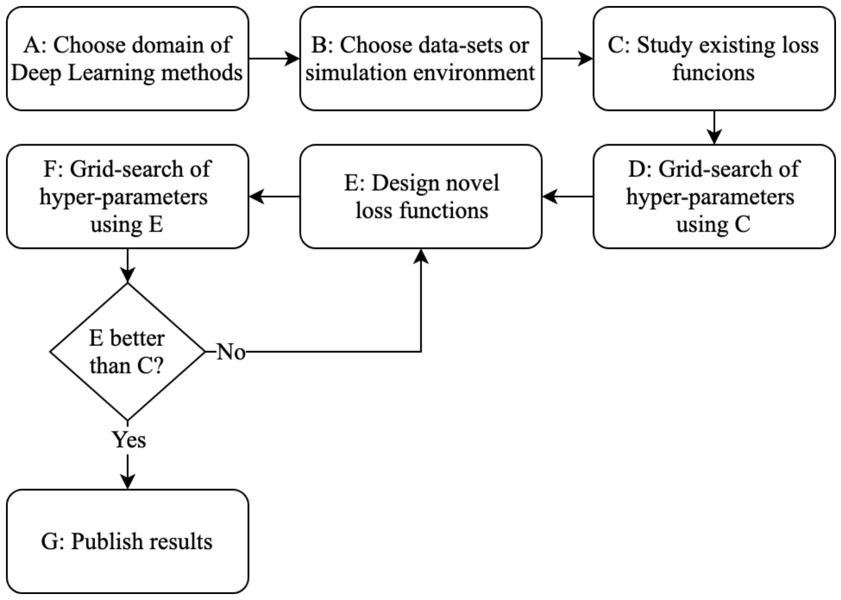

A: Choose domain of Deep Learning methods
B: Choose data-sets or simulation environment
C: Study existing loss funcions
D: Grid-search of hyper-parameters using C
E: Design novel loss functions
F: Grid-search of hyper-parameters using E
E better than C?
No
Yes
G: Publish results

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 was 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 most of the 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] has 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 descriptions of: a novel loss function MDQN for Deep Reinforcement learning in Subsection 3.3, a novel loss function, Exponential Triplet loss for Deep Metric learning in Subsection 4.3, 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 in Subsection 2.4, 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 but the co-authors did research in the chemistry field.

The research work included in the Thesis has been published in the proceedings of 3 scientific conferences and in 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 Summary of the Doctoral Thesis

The Summary of the Doctoral Thesis comprises of 73 pages. It is divided into eight main sections. The full 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 summary of the doctoral thesis:

Section 1 Introduction of the Thesis describes the research background, research motivation, and research objectives.

Section 2 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.

Section 3 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.

Section 4 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.

Section 5 Presents the experimental results of a novel model and loss function described in the previous sections as well as practical applications where these models and loss functions have already been applied successfully.

Section 6 Describes future research topics discovered by studying novel models and loss functions described in the Thesis.

Section 7 Summarizes the main contributions of this study.

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

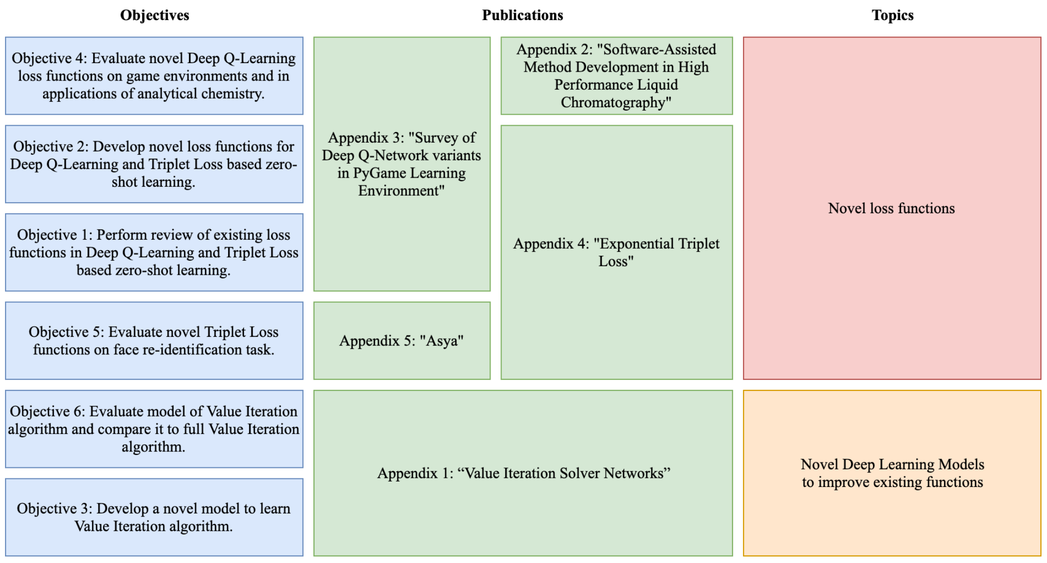

Objectives
Publications
Topics
Objective 4: Evaluate novel Deep Q-Learning loss functions on game environments and in applications of analytical chemistry.
Objective 2: Develop novel loss functions for Deep Q-Learning and Triplet Loss based zero-shot learning.
Objective 1: Perform review of existing loss functions in Deep Q-Learning and Triplet Loss based zero-shot learning.
Objective 5: Evaluate novel Triplet Loss functions on face re-identification task.
Objective 6: Evaluate model of Value Iteration algorithm and compare it to full Value Iteration algorithm.
Objective 3: Develop a novel model to learn Value Iteration algorithm.
Appendix 3: "Survey of Deep Q-Network variants in PyGame Learning Environment"
Appendix 2: "Software-Assisted Method Development in High Performance Liquid Chromatography"
Appendix 4: "Exponential Triplet Loss"
Appendix 5: "Asya"
Appendix 1: "Value Iteration Solver Networks"
Novel loss functions
Novel Deep Learning Models to improve existing functions

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

The list of publications included in the full doctoral thesis document and the main contributions:

1. **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 and a novel synthetic dataset generator OccupancyMapGenerator for evaluation of path planning models.

2. **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.

3. **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.

4. **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.

5. **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 IMPROVING THE PERFORMANCE OF FUNCTIONS USING DEEP LEARNING MODELS

This section of the summary of the doctoral 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 Subsection 2.1, then the existing Deep Learning based methods that can be used to model Value Function are shown in Subsection 2.2 and in Subsection 2.3. Finally, a novel method to model Value Function has been presented in Subsection 2.4. The results of these methods have been shown in Subsection 5.1.

2.1 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 by generating an occupancy grid.

VI is an iterative algorithm that repeatedly applies the same Value function of Equation (3) over all positions of a map to find the cumulative value of each cell position, as shown in Fig. 3.

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 Equation (3).

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.

$$|\frac{\overline{V(s)_i}}{\overline{V(s)_{i-1}}}| < \delta \tag{2}$$

Convergence 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, as shown in Equation (2). 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.

$$V_{i+1}(s) := \max_a \{\sum_{s'} P_a(s,s')(R_a(s,s') + \gamma V_i(s'))\} \tag{3}$$

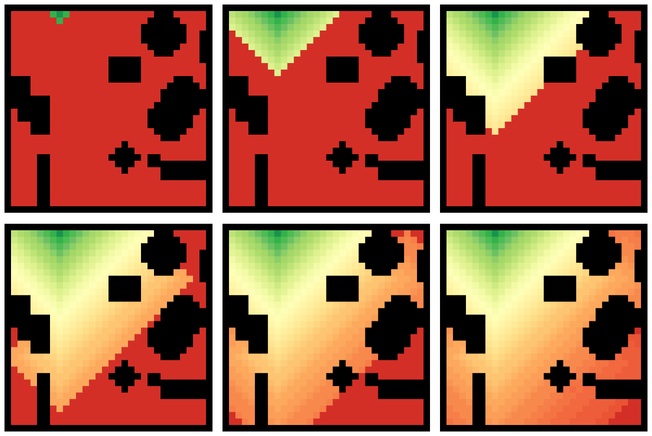


Figure 3: Visualization of the Value Iteration algorithm's consecutive iterations. Iterations depicted from the top left corner to the top right corner, then from the bottom left corner to the bottom right corner. After convergence has been achieved, it is possible to derive the optimal policy from every state to reach the terminal state with the highest cumulative reward. Green colour represents the highest cell values (closest to the positive terminal state) and red colour represents the lowest cell values.

The rationale behind modelling 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 Table 1 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 1:

Comparison of proposed Value Iteration 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

2.2 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 Fig. 4. 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 Fig. 5.

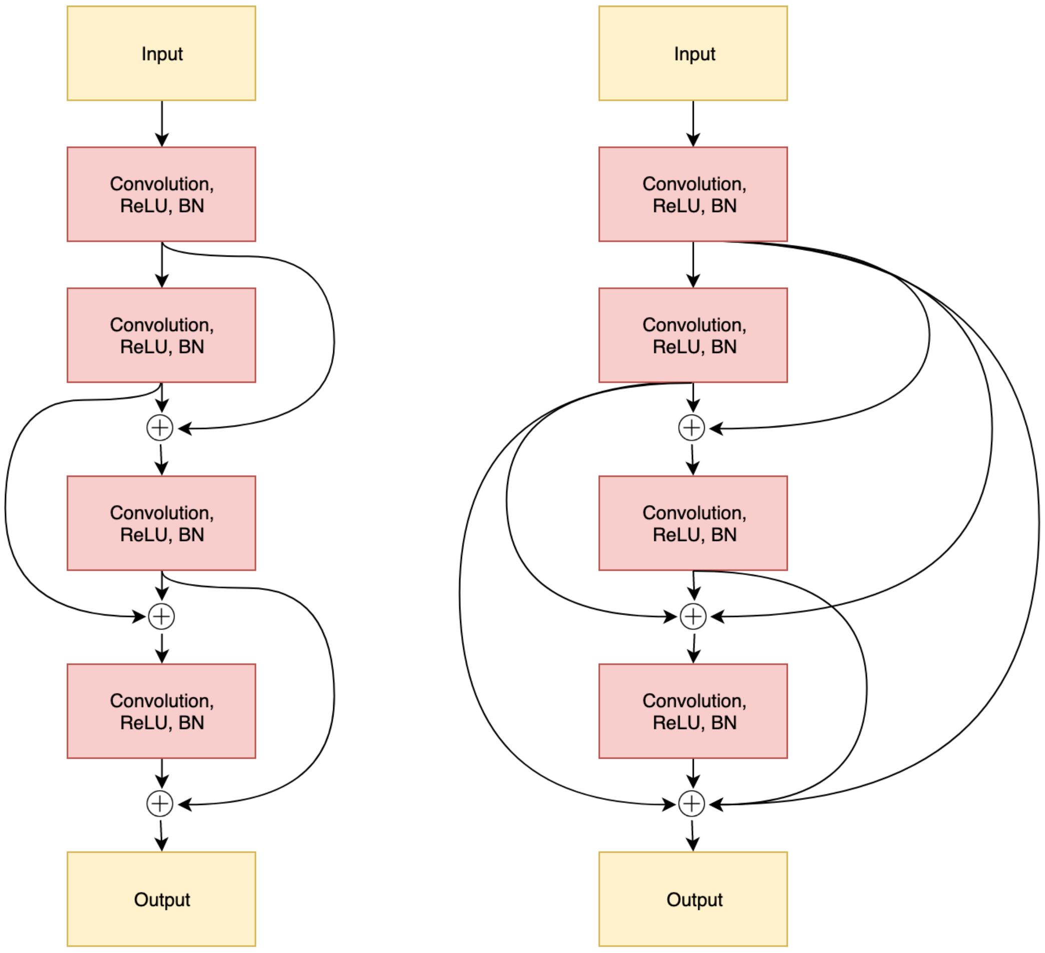


Figure 4: Comparison between ResNet on the left and DenseNet on the right encoder models. Plus sign denotes the arithmetic addition operation.

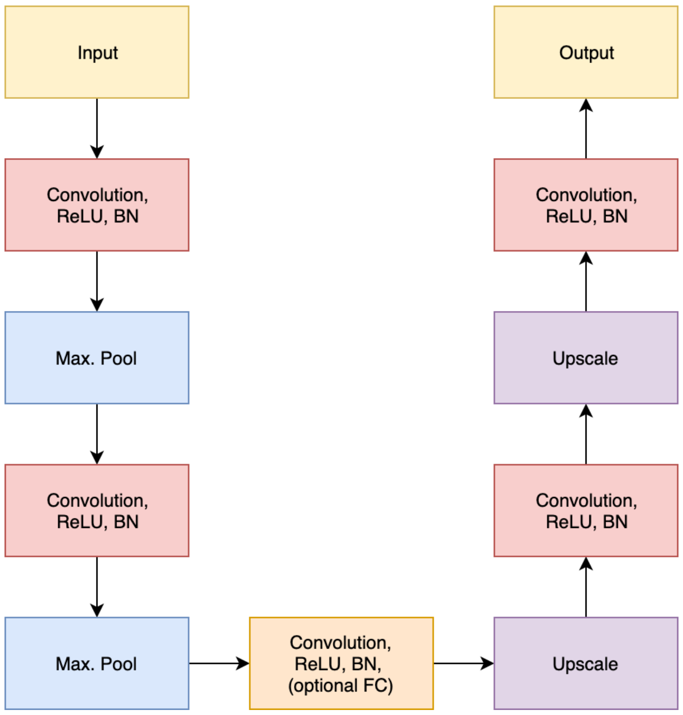


Figure 5: Example of Auto-Encoder model architecture. In the middle of the function graph, an optional affine transformation (FC) could be added if the models have fixed sized inputs and outputs.

The behaviour 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.

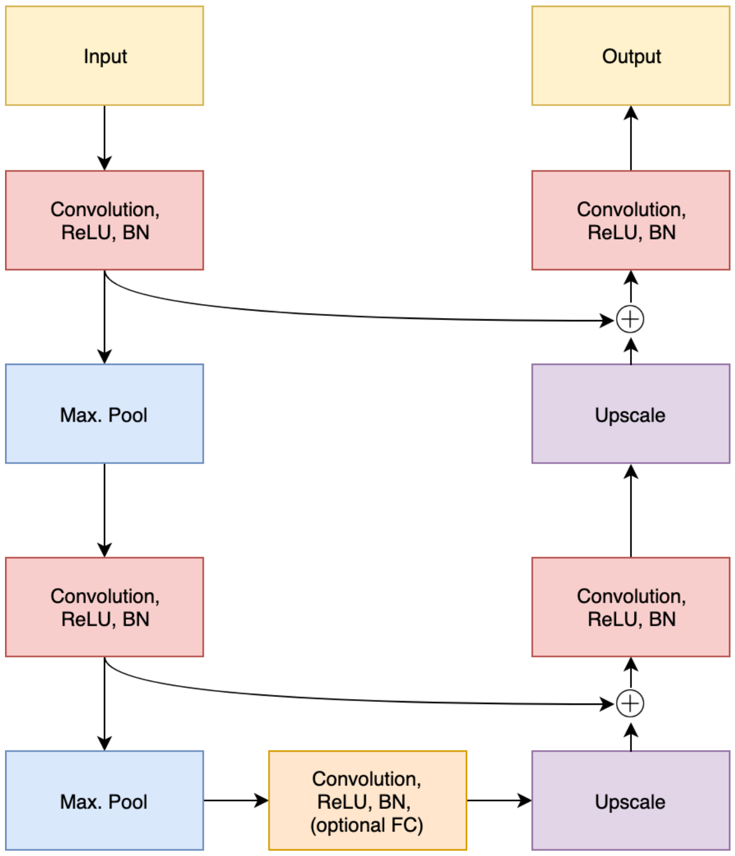


Figure 6: Example of UNet model architecture. Using skip connections, features are added from the encoder part of the function graph to the decoder part.

The original UNet model, as depicted in Fig. 6 contains the graph of functions, as shown in equation Equation (4). 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 Equation (10) and Equation (13) 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.

$$o_1 = ReLU(Conv(x)) \tag{4}$$
$$o_2 = MaxPool(o_1) \tag{5}$$
$$o_3 = ReLU(Conv(o_2)) \tag{6}$$
$$o_4 = MaxPool(o_3) \tag{7}$$
$$o_5 = ReLU(Conv(o_4)) \tag{8}$$
$$o_6 = DeConv(o_5) \tag{9}$$
$$o_7 = (o_6, o_3) \tag{10}$$
$$o_8 = Dropout(ReLU(Conv(o_7))) \tag{11}$$
$$o_9 = DeConv(o_8) \tag{12}$$
$$o_10 = (o_9, o_1) \tag{13}$$
$$o_11 = ReLU(Conv(o_10)) \tag{14}$$
$$y = \sigma(o_11) \tag{15}$$

2.3 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 minus 1 to achieve the same goal [41], [54], [25], [106].

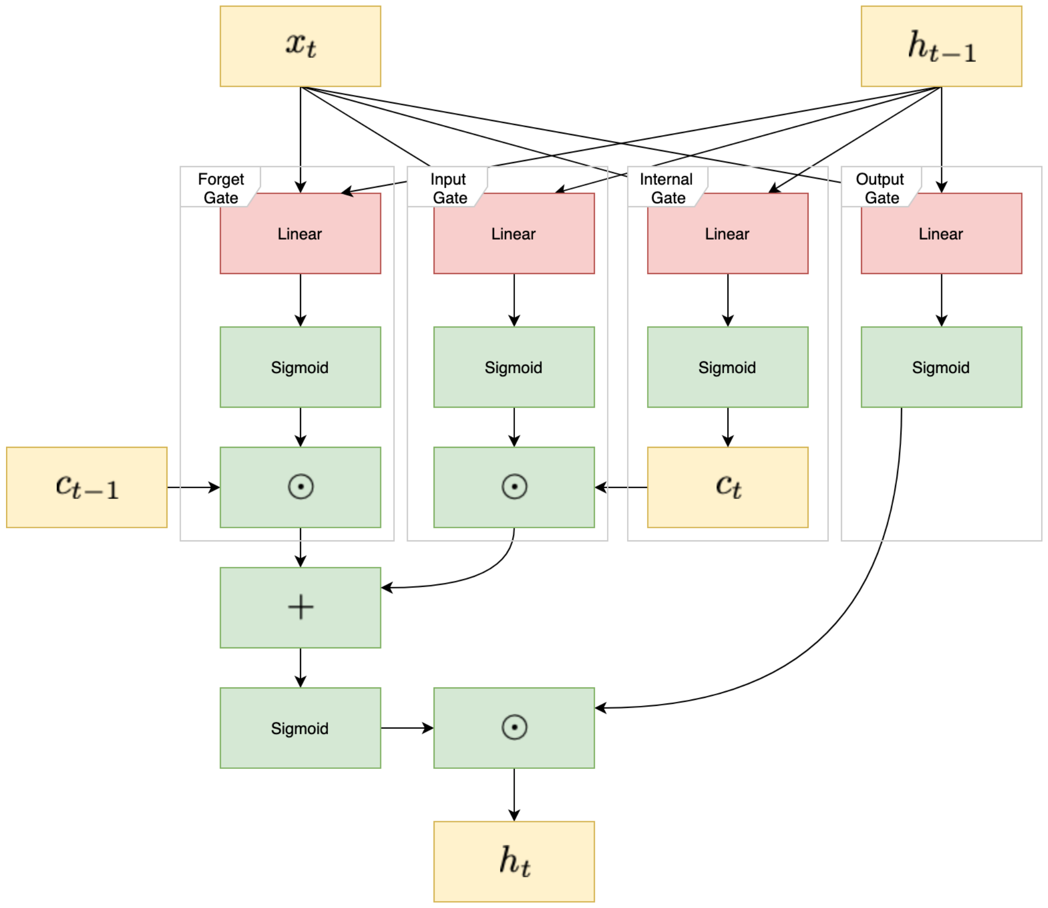


Figure 7: Function graph of LSTM model.

Function graph of LSTM model's equations is given in Equation (17) and visualized in Fig. 7. Trainable weights and biases are denoted with W, U and b, sigmoid function is σ based gates are f_t forget gate, i_t input gate and o_t output gate. Internal state is 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].

$$h_t = \sum_{k=i}^{layers} h_t^k \tag{16}$$

Layer Normalization should be added for all linear transformations within LSTM cell [3] and, finally, multiple LSTM cells should be stacked upon each other and should sum together like in Equation (16) and as it has been done in ResNet for equal error propagation through all layers [102], [38], [106].

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{17}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{18}$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{19}$$
$$\tilde{c}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \tag{20}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{21}$$
$$h_t = o_t \odot \sigma_h(c_t) \tag{22}$$

GRU, as shown in Equation (23) and Fig. 8, 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 improvements as listed above for LSTM applies also to GRU, except b_r should be initialized to $b_r = -1$ [106].

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{23}$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{24}$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{25}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{26}$$

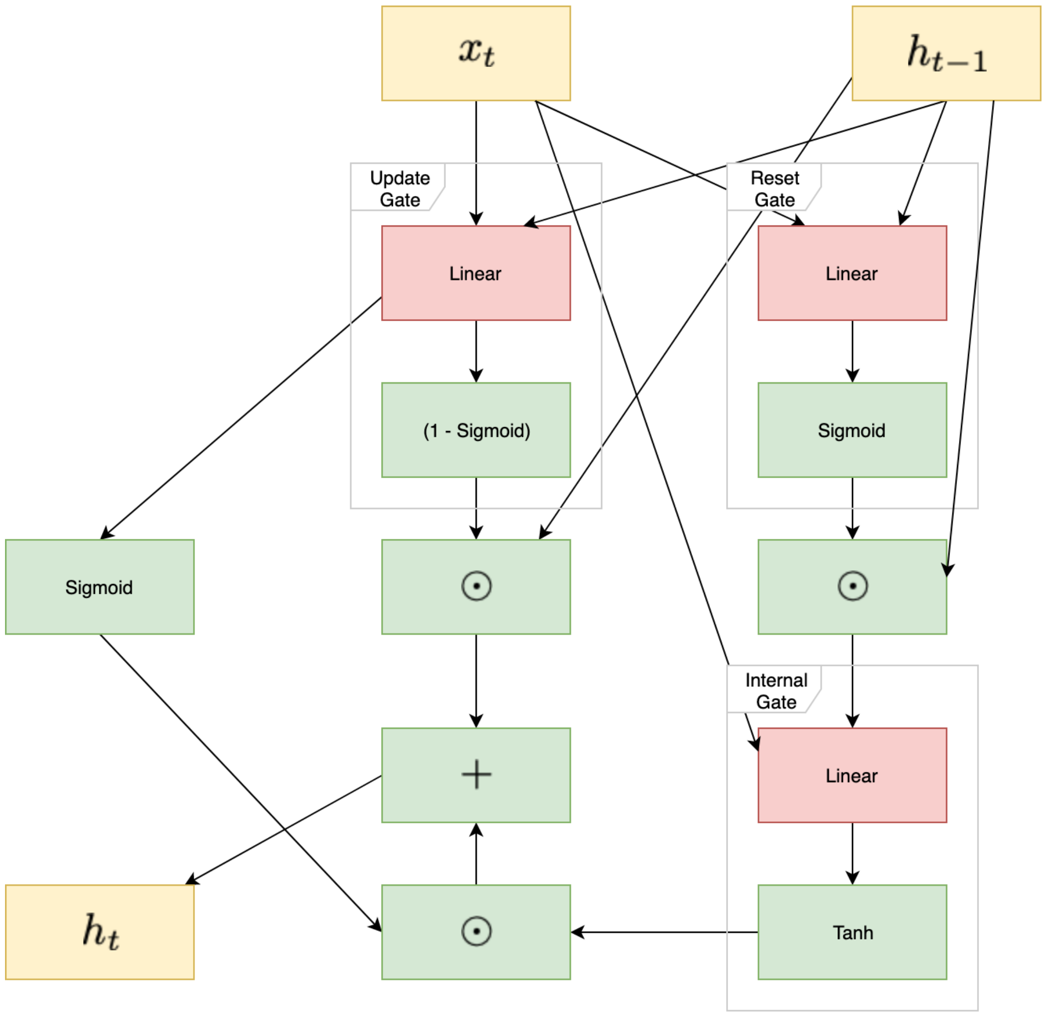


Figure 8: Function graph of GRU.

2.4 UNet-RNN-Skip Model

As a part of the Thesis, a novel UNet and RNN based model has been introduced. The 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 Subsection 5.1. Experimental results of properties of the model are also listed in the same subsection.

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 [31], UNet++ [136] and UNet 3+ [39]. Unlike a simple UNet depicted in Fig. 6 that uses concatenation, the new architecture uses arithmetic addition operations for skip connections. The ResNet [31] function blocks depicted in Fig. 9 have been configured in two different ways. First, there is an Identity ResBlock shown in the equation following Equation (27)

$$y_{c\times w\times h} = ReLU(Conv3x3(x_{c\times w\times h})) \tag{27}$$
$$y'_{c\times w\times h} = BatchNorm(y_{c\times w\times h}) \tag{28}$$
$$z_{c\times w\times h} = ReLU(Conv3x3(y'_{c\times w\times h})) \tag{29}$$
$$z'_{c\times w\times h} = z_{c\times w\times h} + x_{c\times w\times h} \tag{30}$$
$$u_{c\times w\times h} = ReLU(z'_{c\times w\times h}) \tag{31}$$
$$u'_{c\times w\times h} = BatchNorm(u_{c\times w\times h}) \tag{32}$$

Second, there is a BottleNeck which is shown in Equation (33). 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.

$$y_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}} = ReLU(Conv3x3(x_{c\times w\times h})) \tag{33}$$

$$y'_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}} = BatchNorm(y_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}}) \tag{34}$$

$$z_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}} = ReLU(Conv3x3(y'_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}})) \tag{35}$$

$$x'_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}} = Conv1x1(x_{c\times w\times h}) \tag{36}$$

$$z'_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}} = z_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}} + x'_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}} \tag{37}$$

$$u_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}} = ReLU(z'_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}}) \tag{38}$$

$$u'_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}} = BatchNorm(u_{n\cdot c\times\frac{w}{n}\times\frac{h}{n}}) \tag{39}$$

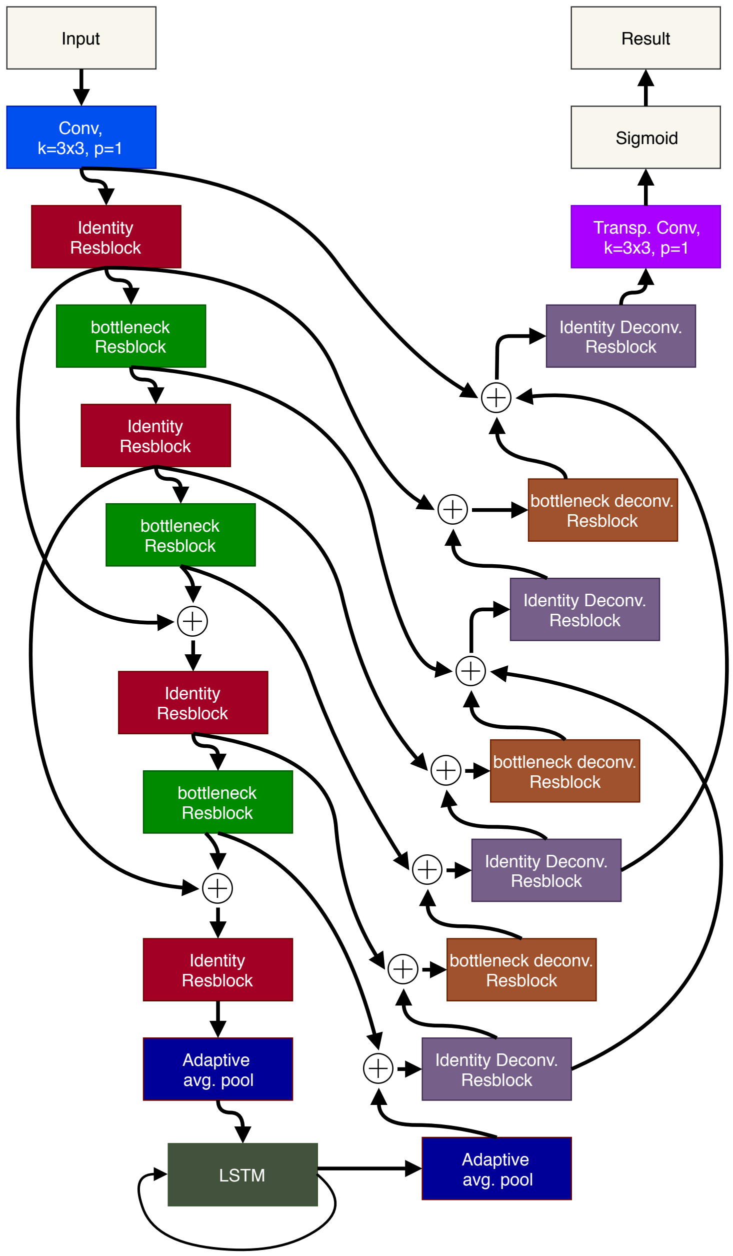


Figure 9: UNet-RNN-Skip model. Colours denote different blocks of functions used by the model.

3 FUNCTION SHAPING IN DEEP REINFORCEMENT LEARNING

This section of the summary of the doctoral 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 Subsection 4.1, then the existing DQN loss function has been described in Subsection 3.2 and finally, a novel loss function has been described in Section **??**. The results of these methods have been shown in Section **??**.

3.1 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, Equation (40) 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 Equation (41).

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.

$$\nabla\pi_\theta(a_t|s_t) = \nabla log\pi_\theta(a_t|s_t)R(\tau) \tag{40}$$

Cumulative reward R Equation (41) is the total reward of the trajectory of states, multiplied at each time step t by 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.

$$R = \sum_{t=0}^{n} \gamma^t r_t \tag{41}$$

The second set of methods is grounded in the Bellman equation Equation (42) that estimates the R 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.

$$Q_\pi(s_t, a_t) = r_t + \max_{a'} Q_\pi(s_{t+1}, a') \tag{42}$$

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.

3.2 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 Fig. 10. 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 behaviour of the environment while the model is trained to better study the properties of the 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) [29], [30] that allows for more stable convergence of DQN based loss function Equation (43) by introducing new DDQN loss function Equation (44) In DDQN model, weights Q_{target} are copied and frozen from Q_Θ every predefined time interval during the process.

$$\mathcal{L}_{dqn} = \begin{cases} (r_t + \gamma \max_{a'} Q_\Theta(s_{t+1}, a') - Q_\Theta(s_t, a_t))^2 & if\ t < t_{last} \\ (r_t - Q_\Theta(s_t, a_t))^2 & if\ t = t_{last} \end{cases} \tag{43}$$

$$\mathcal{L}_{ddqn} = \begin{cases} (r_t + \gamma \max_{a'} Q_{target}(s_{t+1}, a') - Q_\Theta(s_t, a_t))^2 & if\ t < t_{last} \\ (r_t - Q_\Theta(s_t, a_t))^2 & if\ t = t_{last} \end{cases} \tag{44}$$

Figure 10: Atari 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 Fig. 11. In the chart comparison of different loss functions has been shown and methods regarding the millions of frames it took to train them and normalized the score across all Atari games. Rainbow-DQN is based on DDQN loss function with the additions mentioned above.

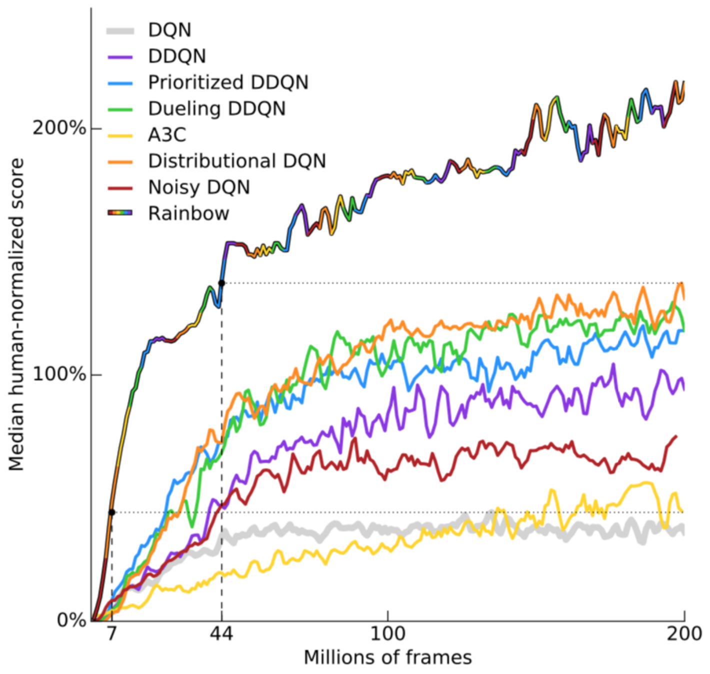


Figure 11: Rainbow-DQN comparison to its parts using normalized Atari's score [35].

3.3 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 behaviour 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 behaviour of MDQN loss function is listed in the pseudocode in Algorithm 1.

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 three 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 the 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 $ReplayBuffer$
```

4 FUNCTION SHAPING IN DEEP METRIC LEARNING

This section of the summary of the doctoral 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 re-identification task is described in Subsection 4.1, the existing Deep Learning loss function to solve the re-identification task is presented in Subsection 4.2 and finally, a novel loss function is explained in Subsection 4.3. The results of these methods are shown in Subsection 5.3.

4.1 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 Fig. 12. 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.

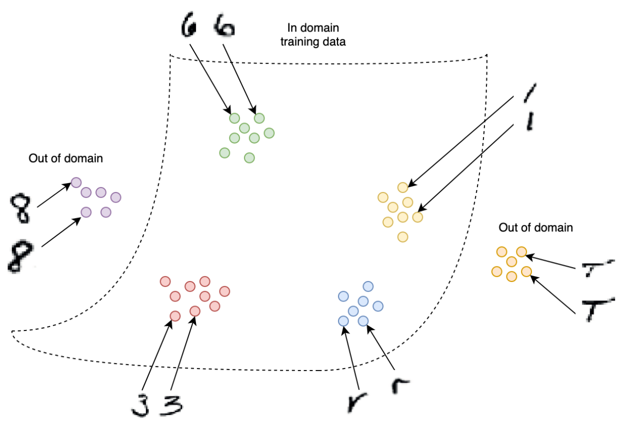
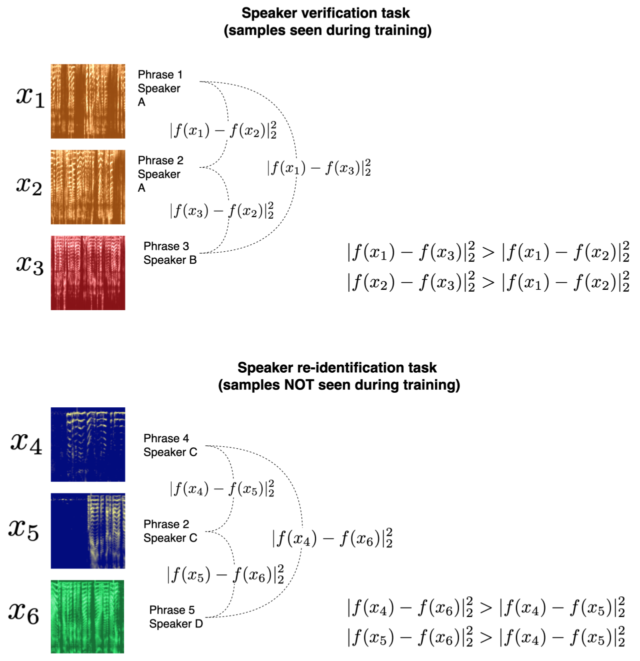


Figure 12: Zero-shot learning visualization using EMNIST dataset with a split of disjoint training and testing datasets.

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 a 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. The 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 re-identification task for face verification and other biometric verification systems [23], as shown in Fig. 13. 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 re-identify this person using any other photo never seen in the enrollment or training process.



Speaker verification task
(samples seen during training)
x_1
Phrase 1 Speaker A
x_2
Phrase 2 Speaker A
x_3
Phrase 3 Speaker B
$|f(x_1) - f(x_2)|_2^2$
$|f(x_1) - f(x_3)|_2^2$
$|f(x_3) - f(x_2)|_2^2$
$|f(x_1) - f(x_3)|_2^2 > |f(x_1) - f(x_2)|_2^2$
$|f(x_2) - f(x_3)|_2^2 > |f(x_1) - f(x_2)|_2^2$
Speaker re-identification task
(samples NOT seen during training)
x_4
Phrase 4 Speaker C
x_5
Phrase 2 Speaker C
x_6
Phrase 5 Speaker D
$|f(x_4) - f(x_5)|_2^2$
$|f(x_4) - f(x_6)|_2^2$
$|f(x_5) - f(x_6)|_2^2$
$|f(x_4) - f(x_6)|_2^2 > |f(x_4) - f(x_5)|_2^2$
$|f(x_5) - f(x_6)|_2^2 > |f(x_4) - f(x_5)|_2^2$

Figure 13: Difference between the speaker verification task and speaker re-identification 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.

4.2 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 Equation (45) 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 Fig. 14. 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.

$$\mathcal{L}_{std} = |\|y_a - y_p\|_2^2 - \|y_a - y_n\|_2^2 + \alpha|_+ \quad (45)$$

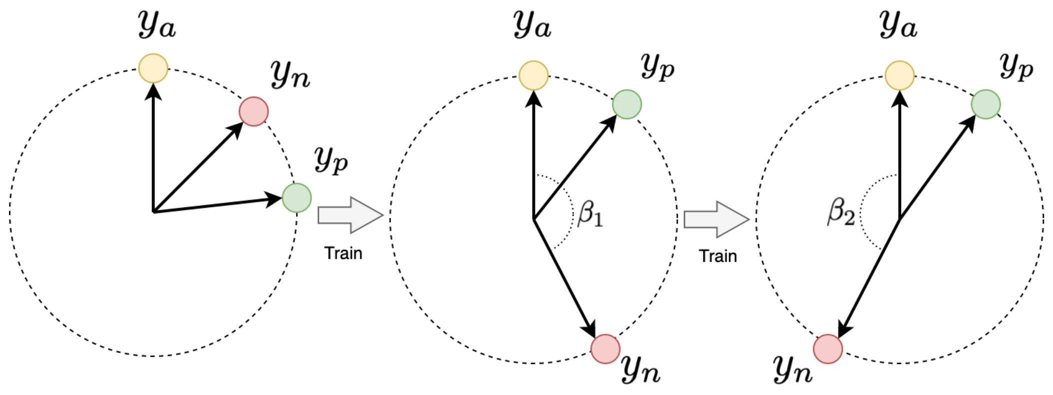


Figure 14: Triplet loss-based training Equation (45) using cosine distances of embedding anchor y_a vector, same class as anchor y_p vector, different class y_n vector. When a negative pair is pushed to a maximum distance, y_n will start to become closer again to y_a.

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 give the largest error gradient and help 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 Appendix D and [110]. Parameters θ are the same for every sample x, as shown in Fig. 15.

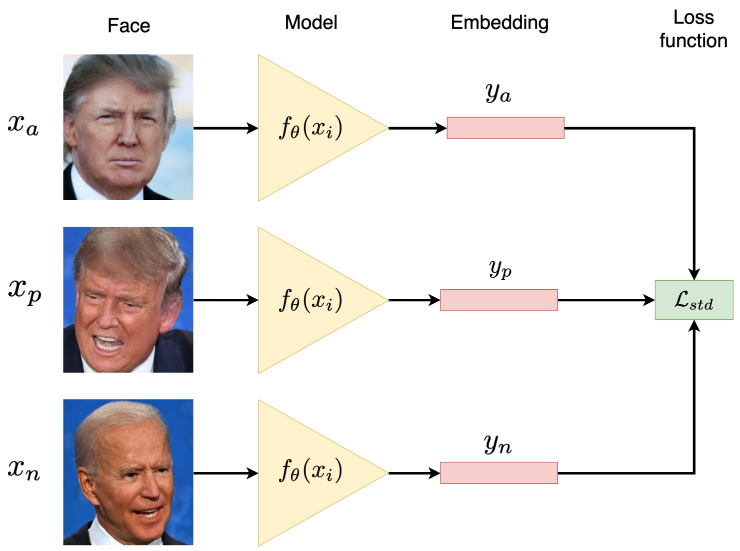


Figure 15: Example of a triplet loss applied for a face re-identification task. Encoder $f_\theta(x_i)$ shares the weights. Dimensionality of y_i is much smaller than x_i.

4.3 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 Fig. 16, 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 keeps 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, as shown in Equation (46).

$$c_n = \frac{\max(f_{emb}(x))}{K} \tag{46}$$

$$emb_p = \frac{\|y_a - y_p\|_2^2}{\max(f_{emb}(x))} \qquad emb_n = \frac{\|y_a - y_n\|_2^2}{\max(f_{emb}(x))} \tag{47}$$

$$\mathcal{L}_{exp} = -log(1.0 - \frac{|emb_p - c_n|_+}{1 - c_n} + \epsilon) - log(1.0 - \frac{|0.5 - emb_n|_+}{0.5} + \epsilon) \quad (48)$$

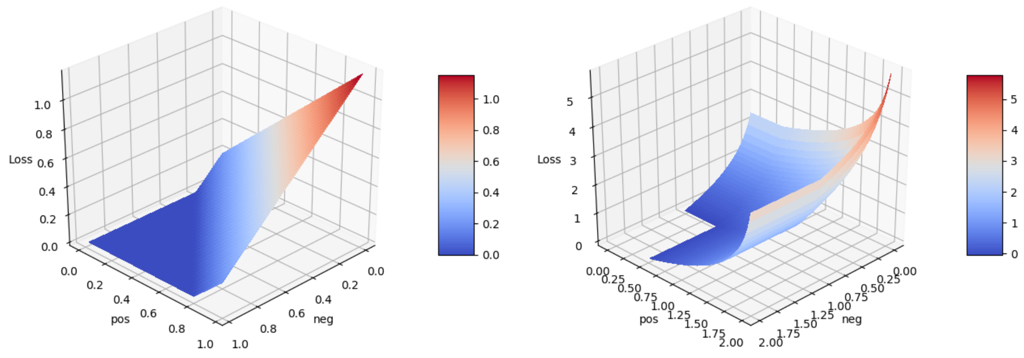


Figure 16: Comparison between $\mathcal{L}_{std}$ and $\mathcal{L}_{exp}$ functions. Positive pair distance $\|y_a - y_p\|_2^2$ (pos) and negative pair distance $\|y_a - y_n\|_2^2$ (neg).

To further improve the performance of L_{exp}, multiple other loss functions from recent research have been combined into the composite loss function *Lcomp*, Equation (52). Additions include L2-constrained Softmax with cross-entropy $\mathcal{L}_{class}$ [84] and center loss $\mathcal{L}_{center}$ [128], Equation (50) and Equation (51) Within $\mathcal{L}_{class}$ Equation (49) 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 centere of the cluster is calculated.

$$\mathcal{L}_{class} = -\sum_{i=1}^{M} y_i log \frac{e^{W_i^T s|f(x_i)|_2^2 + b_i}}{\sum_{j=1}^{C} e^{W_j^T s|f(x_i)|_2^2 + b_j}} \quad (49)$$

$$\mathcal{L}_{center'} = \sum_{i=1}^{M} ||x_i - c_{y_i}||_2^2 \quad (50)$$

$$\mathcal{L}_{center} = \sum_{i=1}^{M} |||x_i - c_{y_i}||_2^2 - \frac{c_n}{2}|_+ \quad (51)$$

$$\mathcal{L}_{comp} = \mathcal{L}_{exp} + C_{center}\mathcal{L}_{center} + C_{class}\mathcal{L}_{class} \quad (52)$$

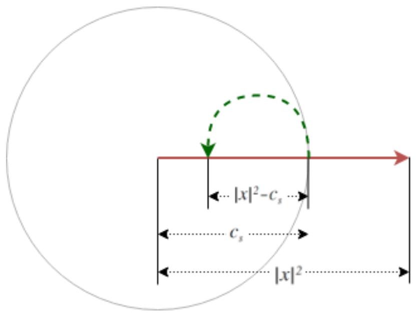


Figure 17: Illustration of Unit-Bounce embedding normalization function within L2 spherical space.

Another contribution to DML is a function to normalize an embedding space called Unit-Bounce, as shown in Equation (53) and Equation (54). It has similar properties to the cosine distance in the Euclidean space. As shown in Fig. 17, when the embedding vector reaches the edge of a sphere, it bounces back towards the centere 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 centere 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.

$$f'_{emb}(x) = \begin{cases} f_{bounce}(x), & \text{if } |x|^2 \geq 1 \\ x, & \text{otherwise} \end{cases} \tag{53}$$

$$f_{bounce}(x) = \begin{cases} |x|^2 - \left\lfloor \frac{|x|^2}{c_s} \right\rfloor - c_s \frac{x}{|x|^2}, & \text{if } \lfloor \frac{|x|}{c_s} \rfloor \bmod 2 = 0 \\ c_s \frac{x}{|x|^2} - |x|^2 - \lfloor \frac{|x|^2}{c_s} \rfloor, & \text{otherwise} \end{cases} \tag{54}$$

5 EXPERIMENTAL RESULTS AND APPLICATIONS

This section of the summary of the doctoral thesis lists the main experimental results of the novel loss functions and methods described in Section 2, Section 3 and Section 4. 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 re-identification 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.

5.1 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 without the need to retrain the VIN for each of the map sizes, as shown in Fig. 18. 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 Table 2

Table 2:

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 3:

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

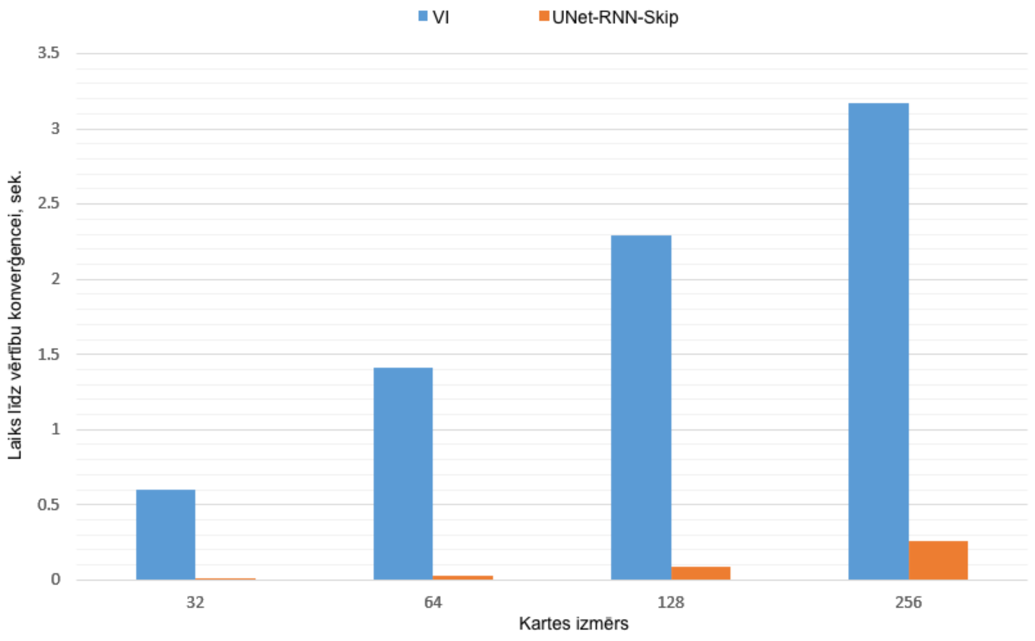


Figure 18: Comparison of convergence speed VI versus VIN based on UNet-RNN-Skip.

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 Fig. 19.

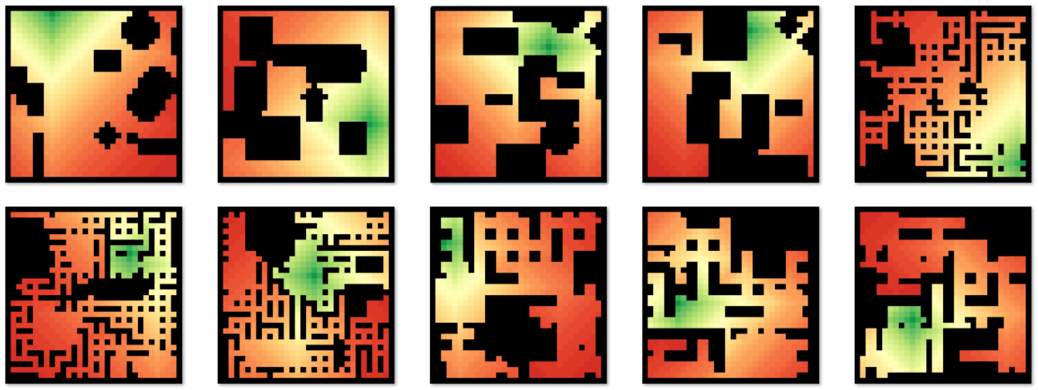


Figure 19: Examples of synthetic maps generated by OccupancyMapGenerator and cell values to reach the positive terminal state (Green - the highest value, Red - the lowest value).

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 in Algorithm 2.

Algorithm 2: OccupancyMapGenerator map generation algorithm

```
1: procedure GenerateMap
2:     $size$
3:     $types\_obstacles = \{maze, circles, rectangles\}$
4:     $max_{coverage}$
5:     $iterations\_obstacles$
6:     $\epsilon_{VI}$
7:     $\underset{size \times size}{M} \leftarrow generateZeros(size)$
8:     if $maze \in types\_obstacles$ then
9:         $M \leftarrow generateMaze(M)$
10:    if $circles \in types\_obstacles$ or $rectangles \in types\_obstacles$ then
11:        for $iterations\_obstacles$ do
12:            $coverage = \frac{walkable}{size^2}$
13:            if $coverage < max_{coverage}$ then
14:                $M \leftarrow generateObstacles(M, types\_obstacles)$
15:            else
16:                break
17:    $goal_{x,y} \leftarrow RandomWalkablePosition(M)$
18:    for $pos_{x,y}$ in $M$ do
19:        if $pos_{x,y} \in walkable$ then
20:            $reachable \leftarrow dijkstra(M, goal_{x,y}, pos_{x,y})$
21:            if $\neg reachable$ then
22:                $M \leftarrow fillHole(M, pos_{x,y})$
23:    $M_{vi} \leftarrow valueIteration(M, \epsilon_{VI})$
24:    $store(M_{vi}, M)$
```

5.2 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 Fig. 20. 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 Table 4, MDQN loss function achieved higher performance than DDQN loss function that at the time of publication was state-of-the-art approach for Deep Q-Learning based reinforcement learning [29].
Grid search of hyper-parameters and 20 repetitions of training procedures were done for all combinations of environments and loss functions to ensure a fair comparison between methods.

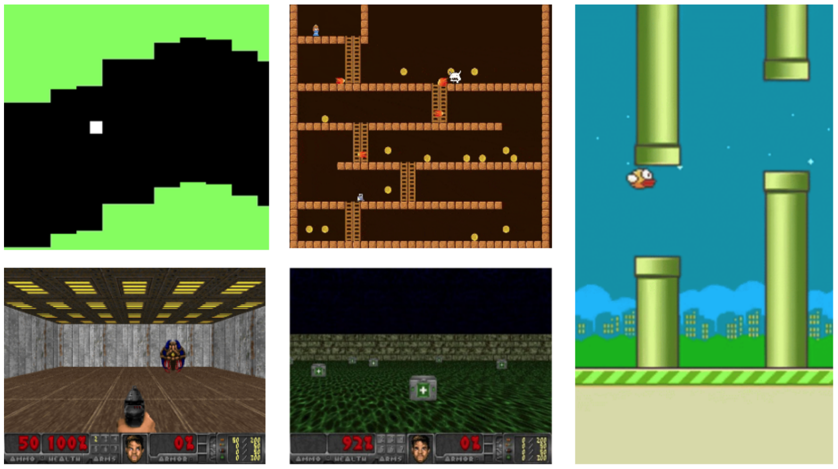


Figure 20: Example of PLE games. Top row from left: "Pixel Helicopter", "Monster Kong". On the right: "Flappy Bird". In the bottom row: 2 variants of VizDoom mini-games that have also been trained within this research (video of an agent that is trained using MDQN loss function: `https://www.youtube.com/watch?v=oqN6rtnv1EI`)

MDQN loss function outperformed DDQN loss function in PLE environments and was more robust when experiments were repeated multiple times, as shown in Table 4.
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 Fig. 21.

Table 4:

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

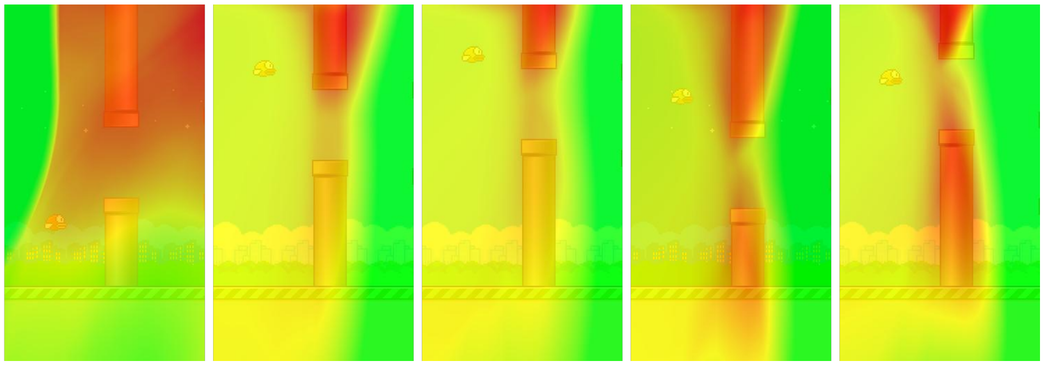


Figure 21: Q-Value Map of FlappyBird environment, where each pixel represents a Q-Value if the player (bird) would be located in this position. Red colour denotes low Q-Value and green colour denotes high Q-Value. On the left there are Q-Value maps before training, in the middle during the training, 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 Fig. 22. As shown in Table 5, 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 5:

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

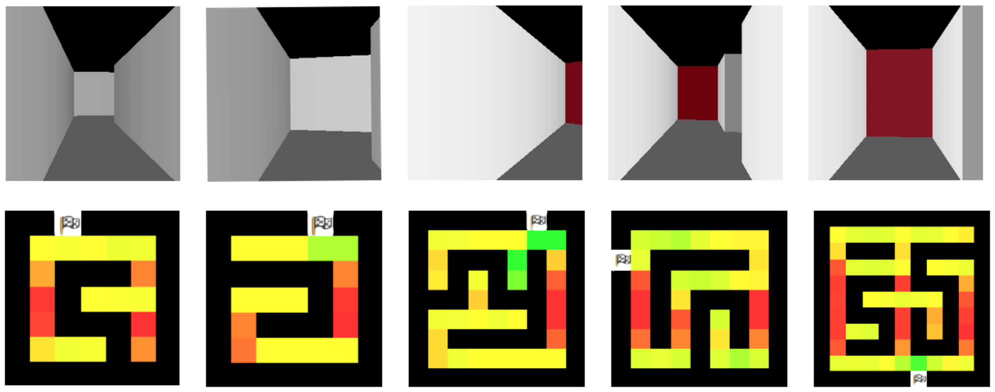


Figure 22: In the top row of frames: 3D raycast maze environment. In the bottom row: Q-Value Map of the environment from top-down view of the 3D raycast maze before, during, and after training.

Finally, along with the novel the 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 Fig. 23, 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.

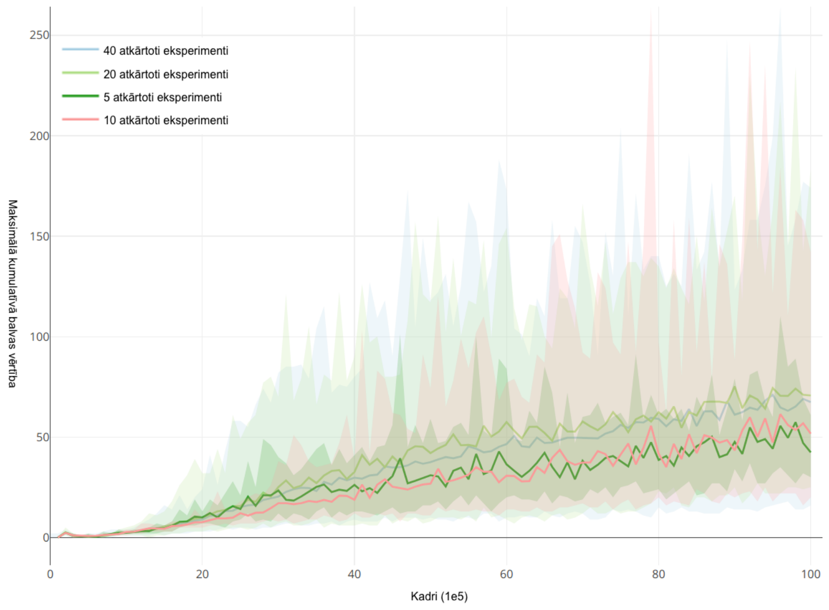


Figure 23: Variance of the score of the same hyper-parameters with different number of re-training procedures of Deep Q-Network in Flappy Bird environment.

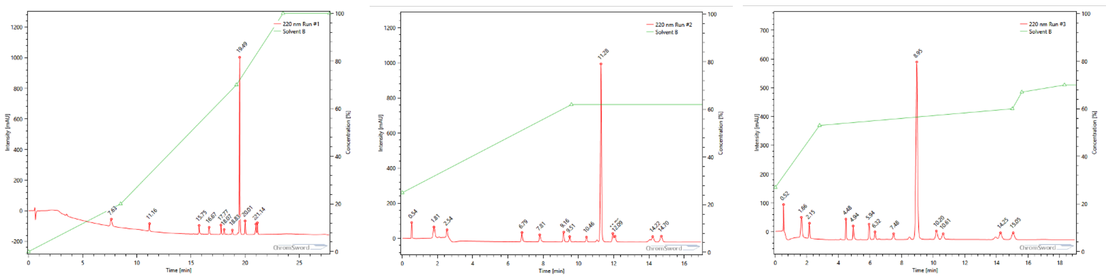


Figure 24: Deep learning-based agent executes consecutive HPLC runs to find the best solvent gradient for peak separation of compounds.

MDQN loss function and other Deep Learning methods have been tested also in a commercial project in SIA ChromSword. As shown in Fig. 24, 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.

5.3 Results of Exponential Triplet Loss Function

Results of Exponential Triplet loss L_{exp} for the class re-identification task are shown in Table 6 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 re-identification 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 re-identification 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 6:

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 re-identification task. The results for zero-shot learning depicted in Fig. 25 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.

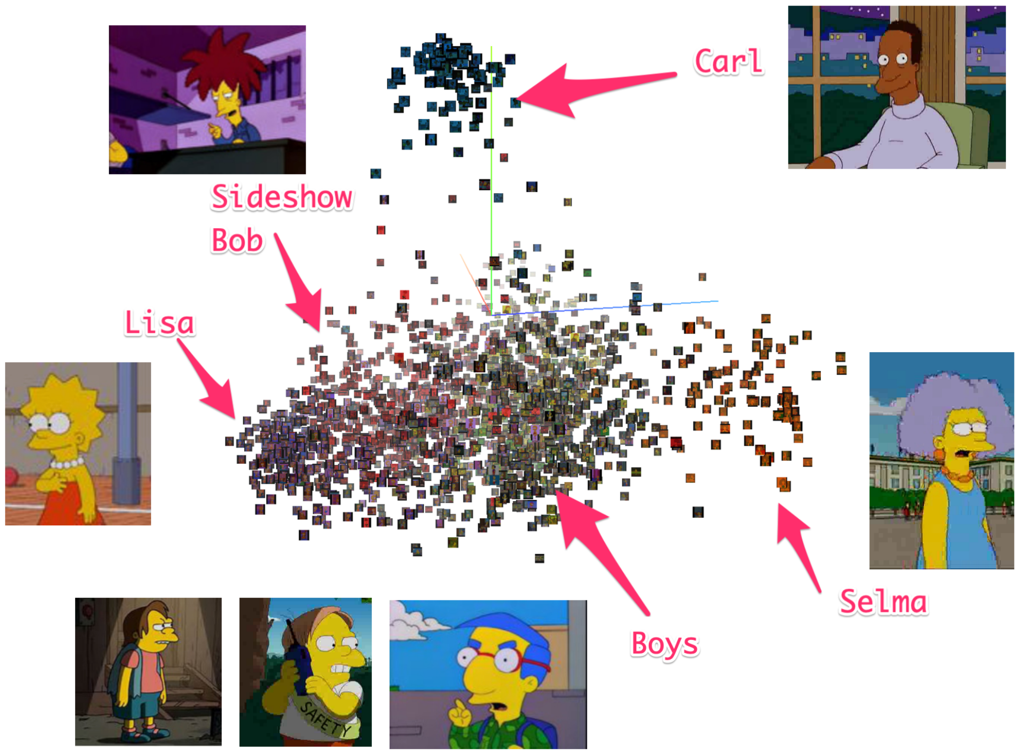


Figure 25: Visualization of PCA of the Simpsons test dataset that Exponential Triplet loss-based model has not used for training.

5.4 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 further.
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]. The 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]. The 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]. The novel model can be used to remove noise and film degradation artifacts, as well as to improve compression of video by reducing complexity in regions that human observer would not pay attention to.

6. Audio denoising tasks [58], [131]. This task has been successfully applied in commercial solutions in SIA Asya. The 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 a novel MDQN loss function has been tested using PLE computer games. MDQN loss function is especially efficient when used 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 in SIA ChromSword. The model is capable of finding the gradient of solvents for compound separation in 2 hours, while before manual sequences of experiments it 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 in 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.

6 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 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 (Kullback—Leibler) 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.

7 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, OccupancyMapGenerator 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 a 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 for 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 Subsection **??** 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 re-identification 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 Subsection 5.4 Applications include zero-shot tasks not only for biometric re-identification, but also for 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

The research has been completed with the support from the IKSA Research Lab and High-Performance Computing Centre of Riga Technical University. RTU HPC provided 12 nVidia K40 GPUs and 8 nVidia V100 GPUs. Special thanks go to Lauris Cikovskis, manager of RTU HPC, and to Professor Agris Nikitenko, the Dean of Faculty of Computer Science and Information Technology and manager of IKSA Research Lab.

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 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`.

[30] 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`.

[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.