

Value Iteration Solver Networks

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

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

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

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

I. INTRODUCTION

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

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

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

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

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

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

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

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

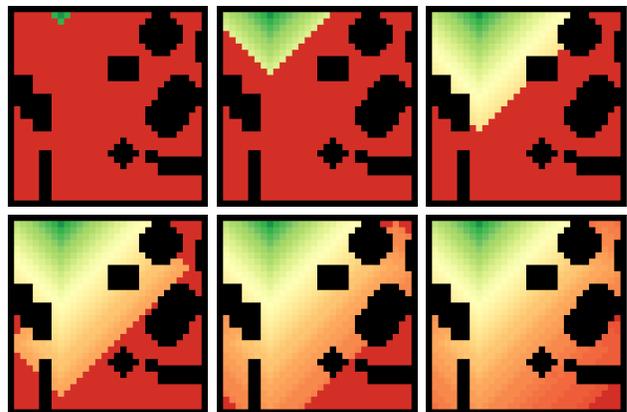


Fig. 1. Visualization of Value Iteration Algorithm's consecutive iterations. After convergence, it is possible to derive optimal policy from every state to reach the terminal state with highest cumulative reward.

II. RELATED WORK

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

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

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

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

III. METHODOLOGY

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

A. UNet variant

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

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

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

B. UNet-RNN variant

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

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

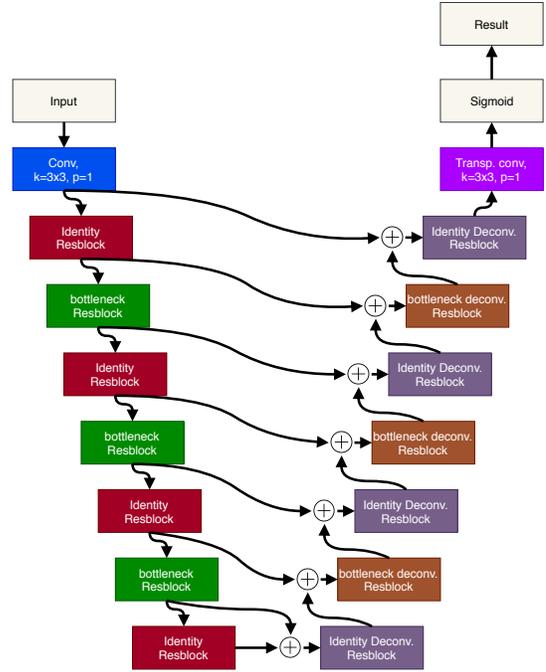


Fig. 2. Residual UNet architecture. Colors denote different building blocks used for model.

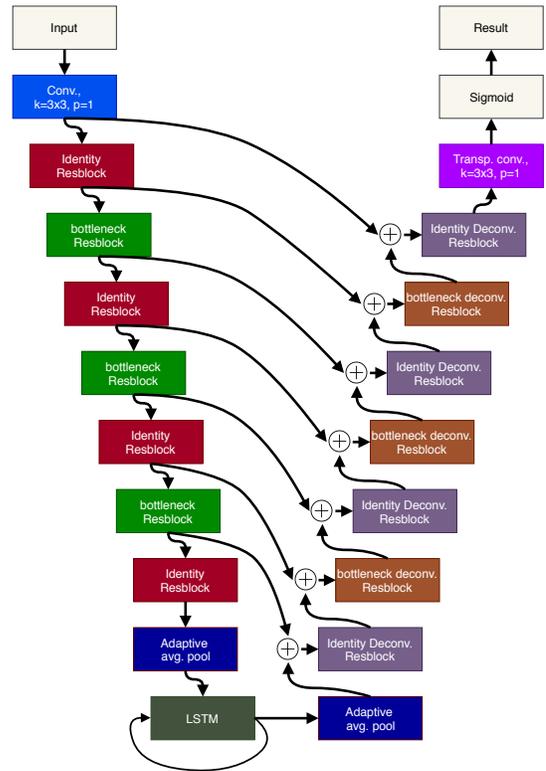


Fig. 3. UNet-RNN architecture. Colors denote different building blocks used for model.

C. UNet-RNN-Skip variant

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

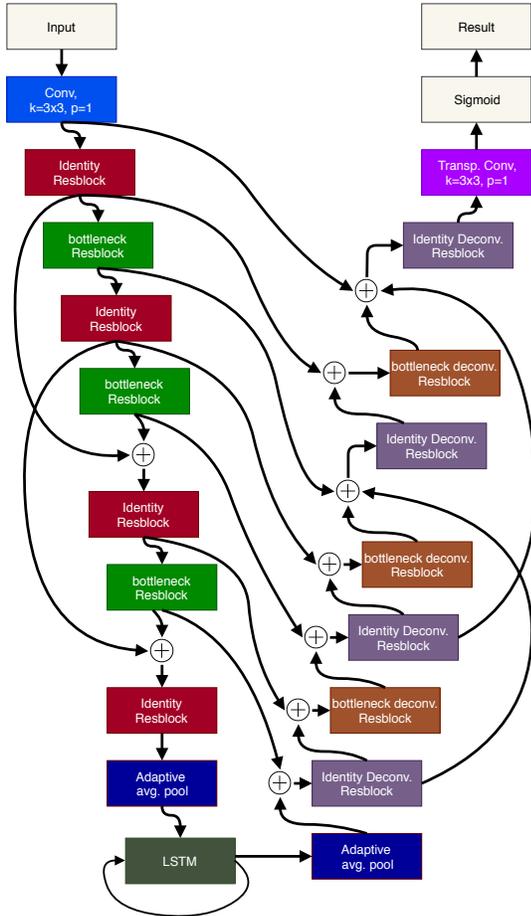


Fig. 4. UNet-RNN-Skip architecture. Colors denote different building blocks used for model.

IV. EXPERIMENTS

A. Data set

While there are some available synthetic data-set generation tools available [8], we created our grid map generator of varying complexity. The grid map generator can generate maps of different sizes and different coverage of obstacles in the map. It is possible to choose between different obstacle types and map types, for example, rectangular obstacles or obstacles generated from the maze patterns Fig. 5. Source code is available

at <https://gitlab.com/VVecins/OccupancyMapGenerator.git>. On synthetically generated grid maps we applied the value iteration algorithm to create the ground truth data-set for the training of the model. For the value iteration algorithm generator, changeable parameters include the map size, the discount value, allowed movement types and environment type (deterministic or stochastic). In this research, iterations were saved when the delta for values between iterations was higher than predefined constant (0.1). Not saving all iterations was for training RNN model to solve the multiple value iterations in one iteration. Source code available at <https://gitlab.com/VVecins/ValueIterationGridmap.git>.

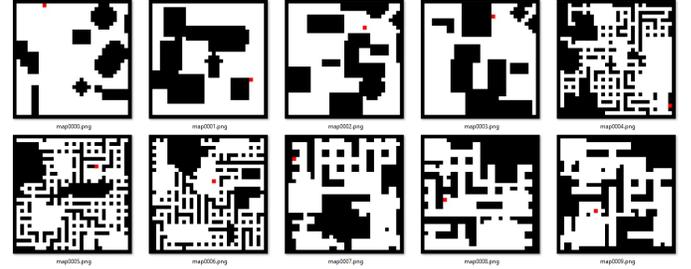


Fig. 5. Examples of synthetic maps generated by OccupancyMapGenerator.

B. Implementation details

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

C. Metrics

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

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

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

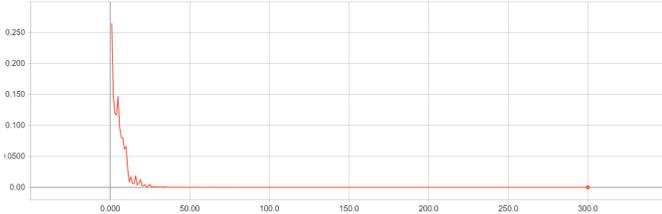


Fig. 6. Mean Squared Error test loss depending on training epoch.

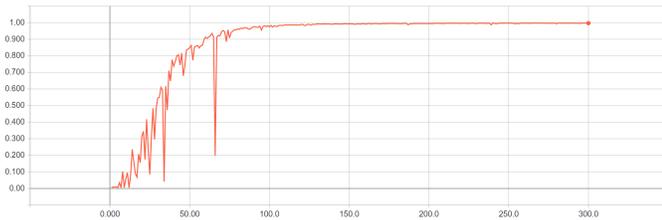


Fig. 7. Success rate metric for test data-set depending on training epoch. Value of 1.0 means that from all states in the map it is possible to reach the the positive terminal state.

D. UNet variant

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

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

E. UNet-RNN variant

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

UNet-RNN, when applied to the map, are changing only a few of the closest grid cells to the wave-front of values. By changing only part of the map and not all of it at the same time, the model can predict better values than models without



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

TABLE I
UNET PERFORMANCE ON 32X32 SIZE MAPS.

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

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

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

recurrent layers. This can be observed by watching activations of layers using Grad-CAM [14] method that allows seeing what parts of the map model gives more attention at different abstraction layers of the model.

F. UNet-RNN-Skip variant

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

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

TABLE III
COMPARISON OF PERFORMANCE OF UNET-RNN MODELS.

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

TABLE IV
UNET-RNN-SKIP MODEL PERFORMANCE.

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

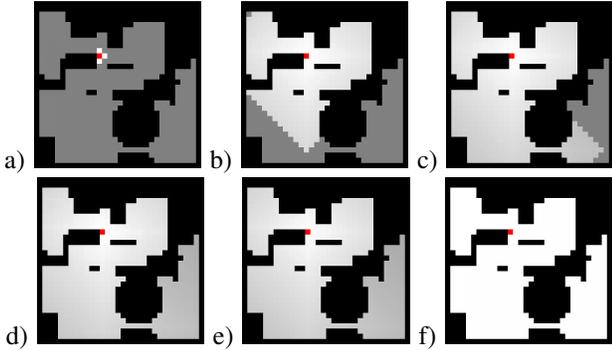


Fig. 9. UNet-RNN-Skip maps generated by progressive iterations (a, b, c), last iteration (d), ground truth (e) and success map (f).

G. Speed of convergence

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

TABLE V
COMPARISON OF EXECUTION TIME IN SECONDS BETWEEN MODELS.

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

H. Experiments on mobile robot platform

We also tested UNet-RNN-Skip model on real-life data gathered from mobile platform using LIDAR as shown in Fig. 11 and Fig. 12. The proposed model can be used for real-

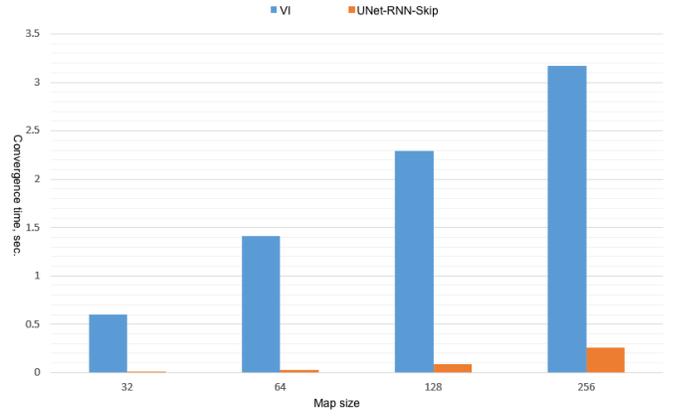


Fig. 10. Comparison of time to convergence between models on different size maps (log scale).

time path planning and obstacle avoidance on GPU powered robotic platforms like nVidia Jetson.

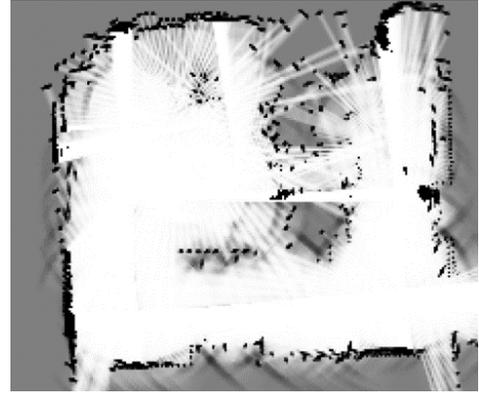


Fig. 11. Map generated from LIDAR point cloud on 2D plane from mobile robot platform.

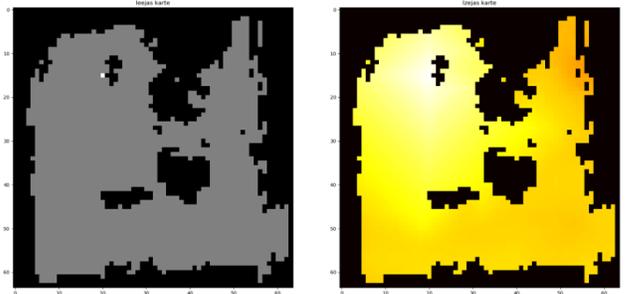


Fig. 12. Discretized map from LIDAR point cloud on 2D plane on the left. The map of values of states after passing through UNet-RNN-Skip model on the right.

V. FURTHER RESEARCH

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

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

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

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

VI. CONCLUSIONS

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

ACKNOWLEDGMENTS

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

REFERENCES

- [1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, Dec. 2002.
- [2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
- [3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2015.
- [4] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," in *AAAI*, 2016.
- [5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
- [6] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.
- [7] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *MICCAI*, 2015.
- [8] A. Tamar, S. Levine, P. Abbeel, Y. Wu, and G. Thomas, "Value Iteration Networks," in *IJCAI*, 2016.
- [9] C. Kamanchi, R. B. Diddigi, and S. Bhatnagar, "Second Order Value Iteration in Reinforcement Learning," *arXiv:1905.03927 [cs, stat]*, May 2019.
- [10] D. Schleich, T. Klamt, and S. Behnke, "Value Iteration Networks on Multiple Levels of Abstraction," *arXiv:1905.11068 [cs]*, May 2019.
- [11] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.
- [12] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.
- [13] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," p. 4.
- [14] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017.