УДС 004.85
DOI: 10.21122/2309-4923-2025-2-4-10

RULKO E.V.

APPLYING ATTENTION U-Net WITH PyTorch ARCHITECTURAL ADD-ONS FOR EXTENSIVE HYPERPARAMETER SEARCH WITH Weights & Biases FOR AREA OF VISIBILITY PREDICTION BASED ON TERRAIN

Military Academy of the Republic of Belarus
Minsk, Republic of Belarus

Abstract. *Current level of development in the sphere of deep learning allows replacing existing domain-specific algorithms for military simulation with approximating neural networks. Hyperparameter search allows finding network's architecture, appropriate for a task. This work describes that process for the task of predicting area of optical visibility, taking a fragment of a digital map as input and proposes ancillary architectural solutions for stitching building blocks together, assuring their conformation for performing search among their possible combinations within the architectural space. The final proposed result is a channel-wise attention U-Net with an encoder, based on ResNet50 backbone.*

Keywords: *: deep learning, U-Net, attention, segmentation, hyperparameter search, W&B, template method*

Introduction

Calculation of an area of potential visibility, based on current terrain and the position of an observer, plays an important role in military simulation. For air defense simulators [1] it's essential to get radar coverage – Figure 1.

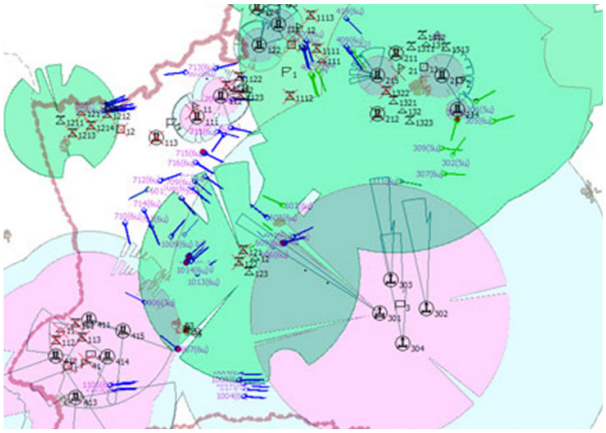


Figure 1. Radar coverage based on terrain

In ground forces simulators [2] every unit must be provided with an area of optical visibility – Figure 2. Building such areas in both cases require moving along azimuth direction with certain discretization and calculating angles of elevation. At the same time current development of deep learning allows tackling such tasks, including calculation of some military domain-specific areas on a map, like artillery shooting range area, area of sustainable communication with the influence of electronic warfare equipment, range of effective fire for specific arms depending on conditions and so on.

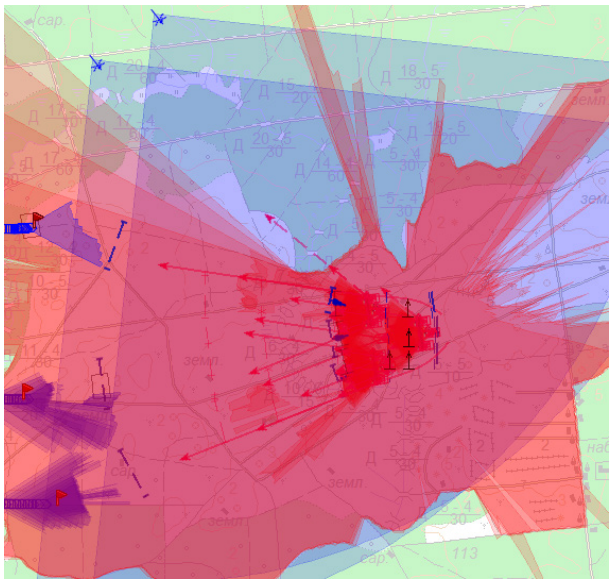


Figure 2. Areas of optical visibility

Transferring military simulation fully into the realm of deep learning, where neural networks are trained on "hardcoded" solutions, will allow building End-to-End systems for military operation planning and utilizing power of generative artificial intelligence for that.

Proposed approach

The area of optical visibility is calculated based on matrixes of height and surfaces for a correspondent patch of area. Usage of the existing military simulation system [2] provides limitless amount of training data. U-Net architecture [3] is widely used for the purpose of producing a mask within the boundaries of the original image,

for task like semantic segmentation. In our case it takes three channels: ideal area, height and surface matrices (U-Net is modified in order to take three input channels) and must produce an area of optical visibility within circumscribed area around a hypothetical observer, situated at the center point. Then the output can be compared with the ground truth area of visibility given from a simulator – Figure 3.

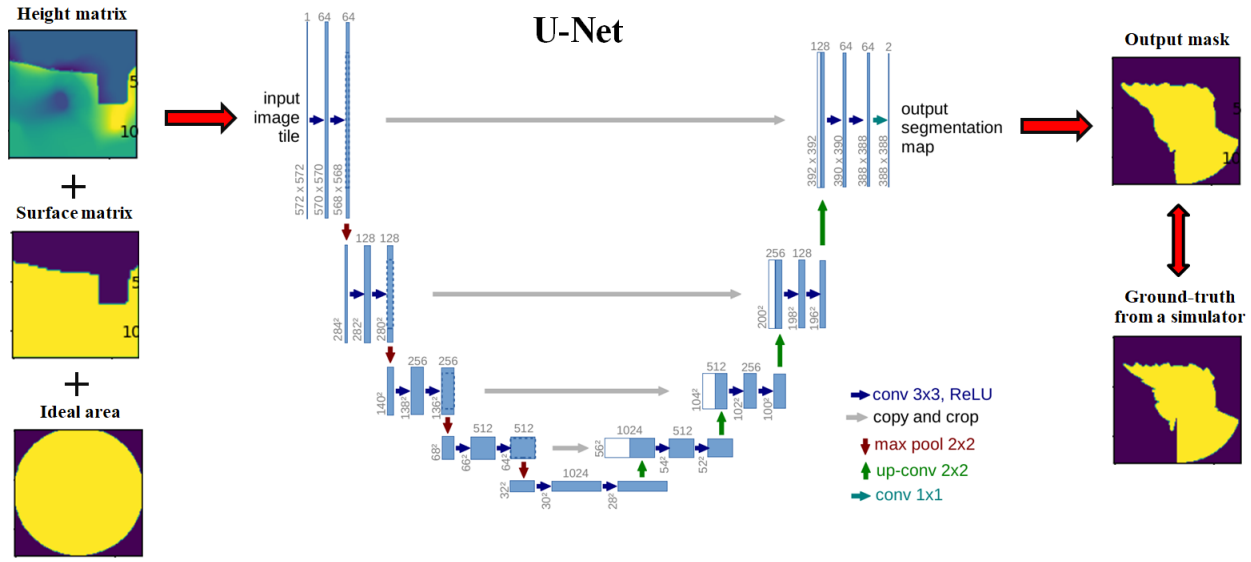


Figure 3. Area of visibility prediction as a mask

However, the usage of U-Net per se, hasn't proved to work well for this. During the research, several approaches were considered, including attention U-Net [4] – Figure 4.

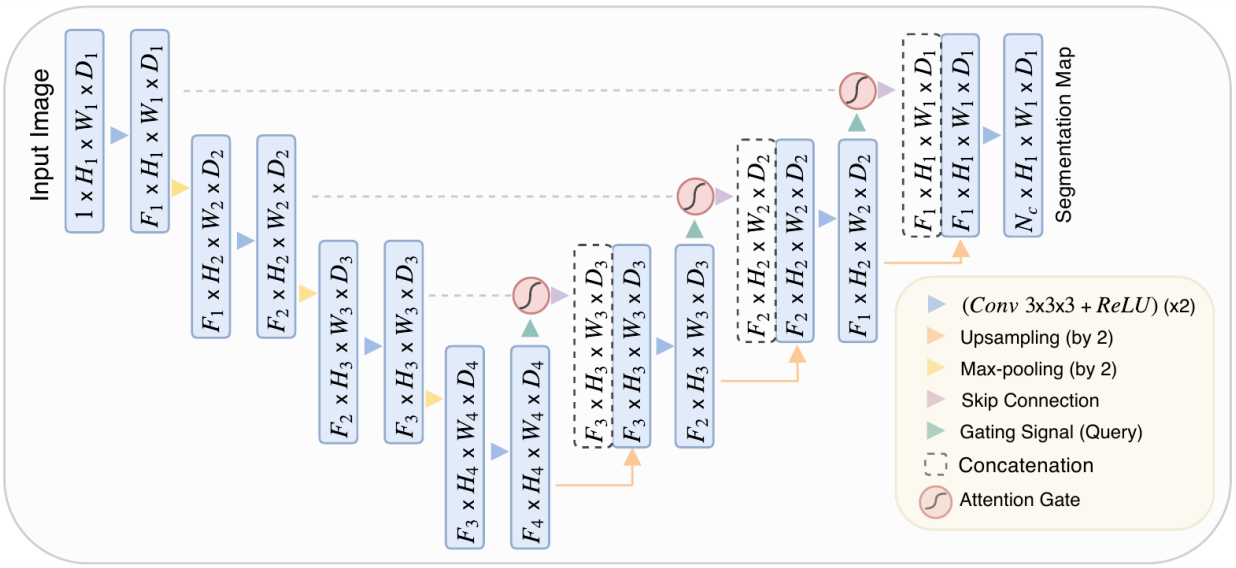


Figure 4. Attention U-Net model [4]

The key idea behind this model is usage of attention gates which learn to suppress irrelevant regions in an input image while highlighting salient features useful for a specific task.

Usage of *Weights & Biases* (W&B) developer platform [5] has proved to be effective for search of optimal hyperparameters such as: learning rate, batch size, dropout values and others. However, search must also be conducted in the architectural space, because of a plethora of possible solutions for particular parts of attention U-Net, like usage of spatial, channel-wise or combined attention; usage of a model that is trained from scratch, like in [6] or usage of a pretrained backbone as an encoder; way of upsampling, like a 2D transposed convolution or a bilinear method; numbers of internal parameters in different attentions blocks and others.

Within W&B sweeps a tested model often must be rebuilt with different architectural parts, instead of just picking a model from list of preconfigured, because of the exorbitant amount of possible combinations. Chosen parts must match each other, like in a case when a number of output channels of a convolutional layer is a tuned hyperparameter, and a consequent fully connected layer must conform that. The following architecture is proposed for assuring compatibility of separate building blocks – Figure 5.

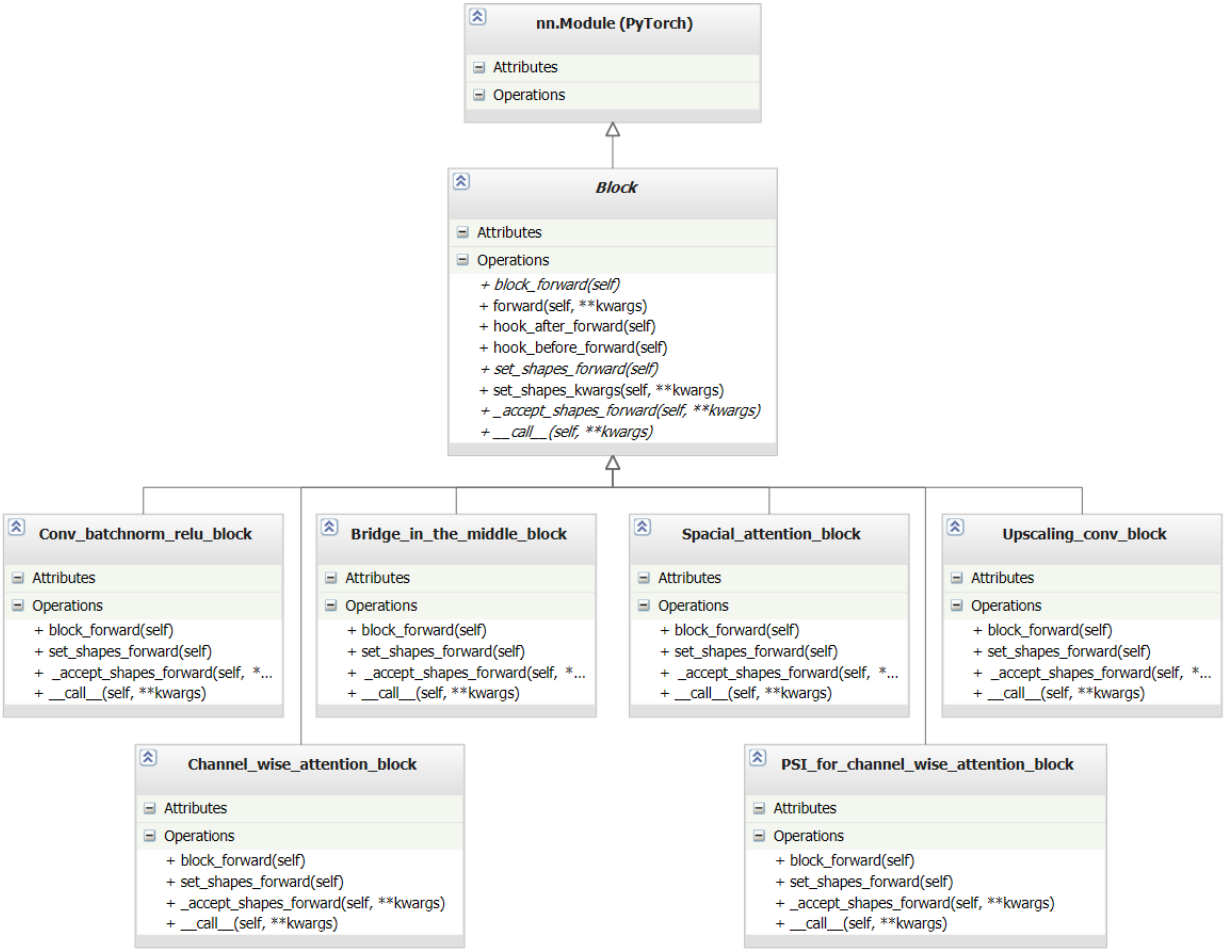


Figure 5. Hierarchy of blocks

An abstract class *Block* is inherited from PyTorch nn.Module and presents a set of abstract methods to be redefined in subclasses. A *forward* method represents a *Template method* pattern [7] which defines a set of consecutive steps (*hook_before_forward, block_forward* and *hook_after_forward*) implemented in subclasses. In the current model architecture some blocks contain others, representing a nested structure, like *Upscaling_conv_block* may contain *Channel_wise_attention_block* which contains *PSI_for_channel_wise_attention_block*. The parameters of nested blocks are set by a higher level block during the first call of *block_forward* through calling *set_shapes_forward* method on nested blocks. Each nested block may have a different signature of parameters to configure, so it in turn calls a not abstract method *set_shapes_kwargs* defined in a *Block* class which checks the matching of number and shapes of passing arguments (just **kwargs*) and calls a method *accept_shapes_forward*, which is implemented by a nested block and has a specific set of parameters in a signature (not just **kwargs*) – Figure 6. Such a gimmick allows having self-descriptive signatures like *accept_shapes_forward* (*self, _gate_plus_x_num_ch: int, _x_channels_number: int*) in *PSI_for_channel_wise_attention_block* class, instead of just passing a dictionary of keywords and parsing them uniquely, depending on a building block. It increases code readability and also allows checking and handling possible mismatch without generating an exception.

Abstract_element_bulider class provides a set of methods for getting building blocks, based on current values from W&B during sweeps for hyperparameter search – Figure 7. Two concrete implementations of it (*WandB_elementbuilder* and *Solitary_elementBuilder*) allow switching between the modes of performing W&B sweeps and manual experiments.

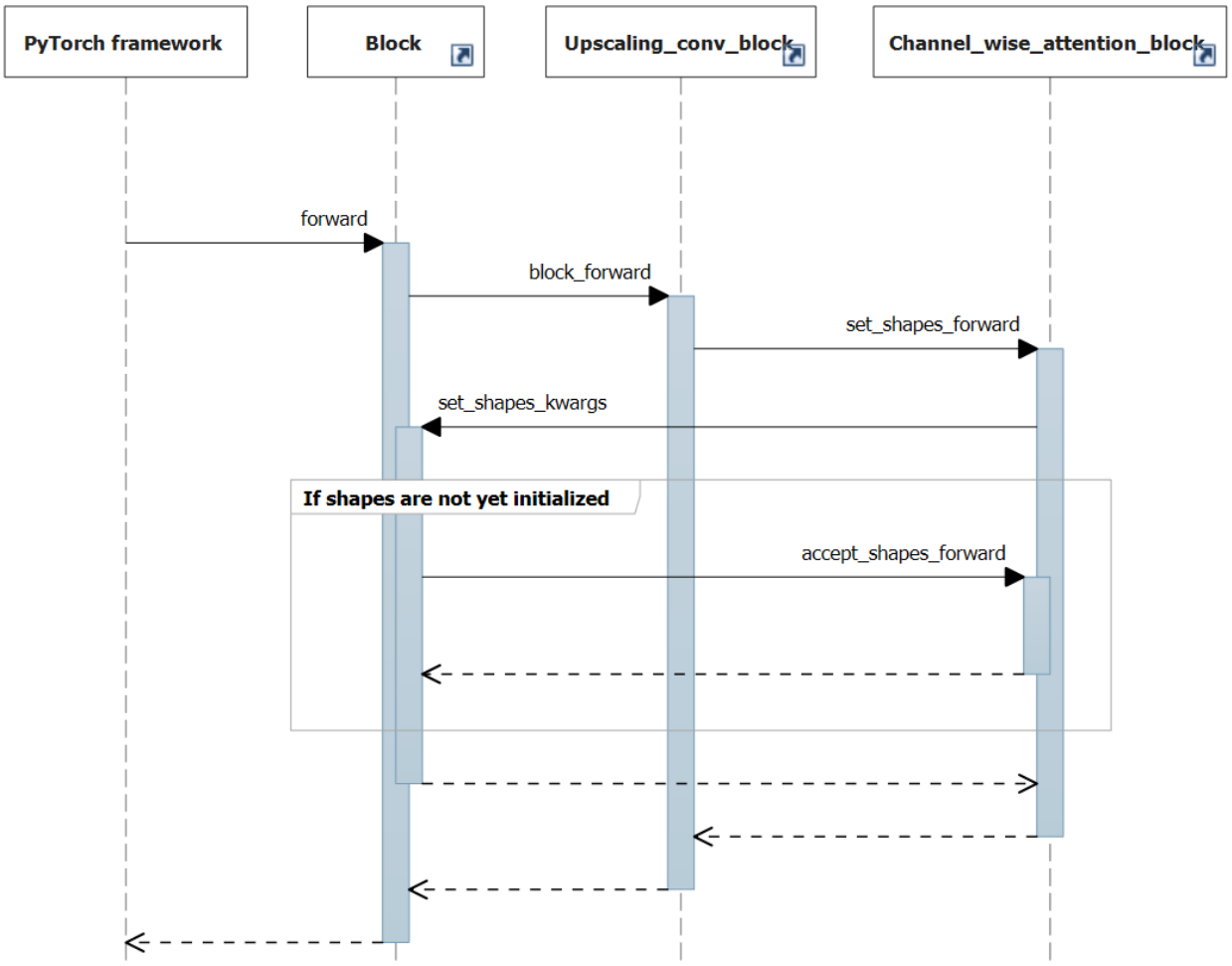


Figure 6. Steps of shape initialization during the *forward* method call

Abstract_train_mode class provides a set of abstract methods for getting parameters, necessary for the training process – Figure 8.

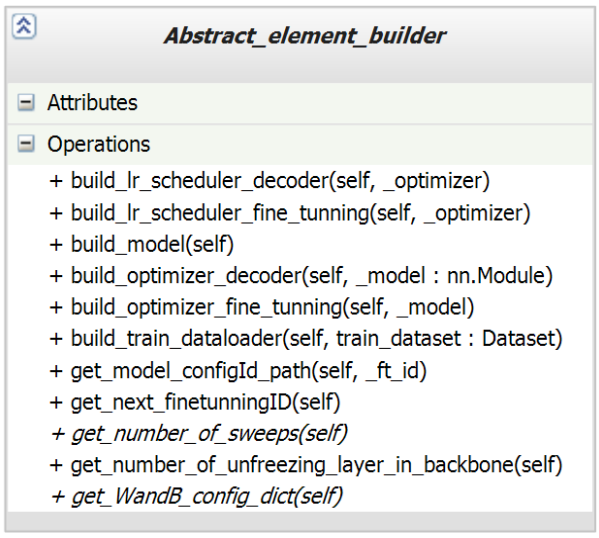


Figure 7. Abstract element builder class

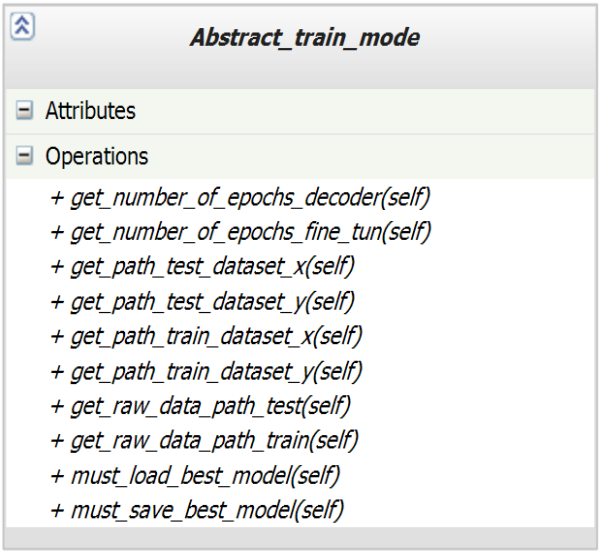


Figure 8. Abstract train mode class

Its main purpose is to support two cases. One of them – End-to-End testing that is provided by

Sanity_check_train_mode implementation, in order not to wait till the end of sweeps with all the training and testing data, like with a *Real_train_mode* implementation, – second case for real training. Another implementation is a *Preliminary_eval_train_mode* class, which is inherited from the *Real_train_mode*. It provides a way of cutting off obviously bad solutions with training on partial amount of data and number of epochs. It's a means of preliminary evaluation before other sweeps with the *Real_train_mode*, and is used as a first phase of parameter search.

During that ***first*** bout of search, it was found out, that a simple U-Net model and a simple attention U-Net model, both written from scratch, works worse than an attention U-Net model, based on a pretrained ResNet50 backbone. Usage of ResNet152 as a backbone improves precision a bit, but at the same time drastically hinders performance in terms of calculation time.

During the ***second*** search iteration, it was determined that channel-wise attention is generally better for the task than spatial attention, as well as that 2D transposed convolution is a better choice for upsampling than a bilinear method.

The ***third*** iteration gave preliminary values for number of internal channels in attention blocks, batch size for decoder training, best optimizer and learning rate. While reaching a descent performance on majority of possible terrains, further analysis of badly predicted cases revealed that a chosen configuration struggles to cope with the case when the observer is situated in the forest near a border with open space, when they can already through the forest – Figure 9 (colors on a surface mask correspond to different surfaces: open space, forest, shrubs).

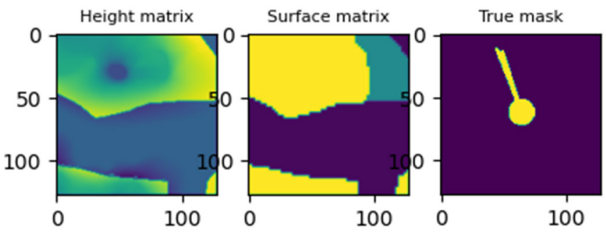


Figure 9. True mask while looking through the forest

Within the frame of research, in order to incentivize the network to understand that it deals with the forest, an auxiliary head with a fully connected layer was added. Its goal is to provide a binary classification: whether an observer is situated in the forest or not, with a contribution to the main binary cross entropy loss (*nn.BCELoss*), whereas Intersection over Union (IoU) is the metric for parameter selection. Schematic representation of the resultant architecture (without attention gates and details) is presented on Figure 10.

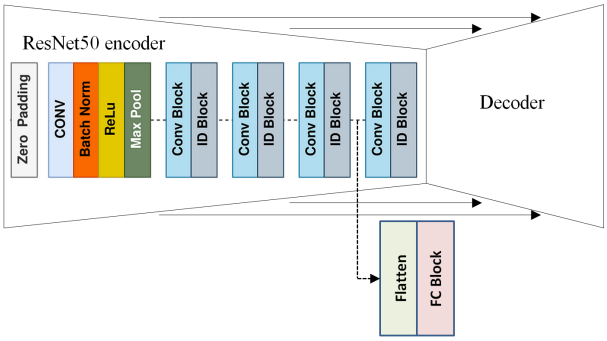


Figure 10. Auxiliary head for binary classification

As a result, such form of feature engineering wasn't successful. The situation when an observer can see through the forest towards open space is quite rare in the train dataset. So, as an aftermath of that, there is class imbalance. There are common solutions to cope with imbalanced datasets like usage of weighted random sampler [8], class weighting [9], synthetic minority oversampling technique (SMOTE) [10], or simple additional oversampling through augmentation of minority samples or some combinations. In our case additional data can be just generated from the simulator, with a stipulated rule, that we add a current case if both requirements are true: the observer is in the forest and there is an open space in the vicinity. After enriching the training dataset, the network started performing much better, which was evident even during preliminary evaluation.

The ***forth*** iteration of hyperparameter search involves usage of the real train mode with full datasets and number of epochs for every considered combination of hyperparameters. It eventually allowed picking a set best hyperparameters for decoder.

The ***fifth*** iteration involves unfreezing encoder and search of hyperparameters for fine-tuning. After that procedure, the network is evaluated on a test dataset, not a validation one used before, in order to check possible overfitting in hyperparameter search. It demonstrates an average IoU of 0.95 on a test set. Some examples of predicted masks (areas of visibility) are presented on Figure 11.

The network was trained on relatively low resolution of 128×128 in order to check the potential possibility of the approach, but in case of a higher resolution it will work the same way.

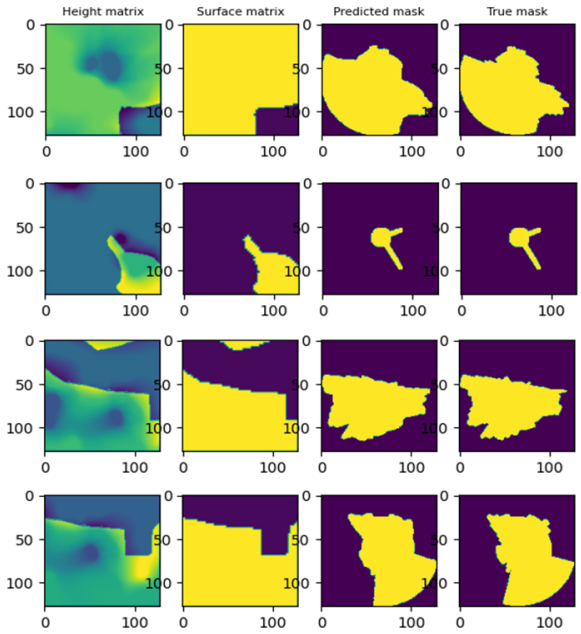


Figure 11. Examples of predictions

Conclusions

Current diversity of neural network architectures makes it possible to utilize them for solving different domain-specific tasks, including building an area of visibility, given data from a digital map. Usage of solutions for hyperparameter search allows evaluating a plethora of possible configurations. Ancillary architectural solutions provide a way for stitching building blocks together, assuring their conformation for performing search among their possible combinations within the architectural space. The final network in this work is based on combination of parts from different solutions, like the attention U-Net form the first source [6], different forms of attention from the second source [11] and finally – concrescence with a popular pretrained backbone in search of best performance. Such a strategy seems to be effective for solving real machine learning tasks for applied science.

REFERENCES

1. **Mathematical model complex for military grouping efficiency assessment:** [Electronic resource] // URL: http://en.belfortex.com/page/show/9. (Date of access: 15/11/2024).

2. **E. Rulko, et al.** Application of a simulation system for optimizing solutions based on elements of the theory of reflexive control. Collection of scientific articles of the Military academy of the Republic of Belarus. 2017. № 32. P. 153–162.

3. **Olaf Ronneberger, et al.** U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. arXiv: 1505.04597.

4. **Ozan Oktay et al.** Attention U-Net: Learning Where to Look for the Pancreas. 2018. arXiv: 1804.03999.

5. **Weights & Biases:** [Electronic resource] // URL: https://wandb.ai/site. (Date of access: 15/11/2024).

6. PyTorch implementation of U-Net, R2U-Net, attention U-Net, attention R2U-Net. https://github.com/LeeJunHyun/Image_Segmentation. 2018.

7. **E. Gamma, et al.** "Design Patterns Elements of Reusable Object-Oriented Software," Addison-Wesley, Massachusetts, 1995.

8. **PyTorch documentation. Weighted random sampler:** [Electronic resource] // URL: https://pytorch.org/docs/stable/data.html#torch.utils.data.WeightedRandomSampler. (Date of access: 15/11/2024).

9. **How to Improve Class Imbalance using Class Weights in Machine Learning?:** [Electronic resource] // URL: https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/. (Date of access: 15/11/2024).

10. **N. V. Chawla, et al.** SMOTE: Synthetic Minority Over-sampling Technique. 2011. arXiv: 1106.1813.

11. **Long Chen, et al.** SCA-CNN: Spatial and Channel-wise Attention in Convolutional Networks for Image Captioning. 2016. arXiv: 1611.05594.

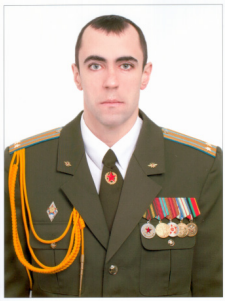
РУЛЬКО Е.В.

ИСПОЛЬЗОВАНИЕ U-Net СЕТИ С МЕХАНИЗМОМ ВНИМАНИЯ СОВМЕСТНО С АРХИТЕКТУРНЫМИ НАДСТРОЙКАМИ ДЛЯ ФРЕЙМВОРКА PyTorch В РАМКАХ ПОИСКА ГИПЕРПАРАМЕТРОВ ПОСРЕДСТВОМ БИБЛИОТЕКИ Weights & Biases ДЛЯ ПРЕДСКАЗЫВАНИЯ ОБЛАСТИ ВИДИМОСТИ ПО КАРТЕ МЕСТНОСТИ

Военная академия Республики Беларусь
г. Минск, Республика Беларусь

Аннотация. *Текущий уровень развития глубокого обучения позволяет заменить нейронными сетями существующие специфические для моделирования военных действий алгоритмы. Поиск гиперпараметров даёт возможность определить структуры сетей, подходящие для решения соответствующих задач. Данная работа описывает процесс поиска структуры сети для предсказания зоны оптической видимости на основе фрагмента цифровой карты местности и предлагает архитектурные решения для комбинирования возможных составных частей сети, обеспечивая их совместимость в рамках поиска наилучшего решения. В качестве финального варианта предлагается использование U-Net архитектуры с поканальным механизмом внимания и энкодером на основе сети ResNet50.*

Ключевые слова: *глубокое обучение, U-Net, механизм внимания, сегментация, поиск гиперпараметров, W&B, шаблонный метод*



Рулько Евгений Викторович, кандидат технических наук, доцент. Начальник научно-исследовательской лаборатории моделирования военных действий учреждения образования «Военная академия Республики Беларусь». Сфера научных интересов: глубокое обучение, машинное зрение, обучение с подкреплением, нейронауки, активный вывод, принцип свободной энергии, рефлексивное управление.

Eugene Rulko, PhD, associate professor in computer science. The head of the research laboratory of military operation simulation of the educational institution «Military academy of the Republic of Belarus». Research interests: deep learning, computer vision, reinforcement learning, neuroscience, active inference, free energy principle, reflexive control.

E-mail: eugeni1533@gmail.com