Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Deepness: Deep neural remote sensing plugin for QGIS

Przemysław Aszkowski, Bartosz Ptak, Marek Kraft [*], Dominik Pieczyński, Paweł Drapikowski

Poznań University of Technology, Institute of Robotics and Machine Intelligence, Piotrowo 3A, 60-965 Poznań, Poland

A R T I C L E I N F O

Article history:
Received 22 May 2023
Received in revised form 14 July 2023
Accepted 2 August 2023

Keywords:
QGIS
Deep learning
Remote sensing
Segmentation
Object detection

A B S T R A C T

This paper presents *Deepness* - an open-source plugin for the QGIS application, allowing the easy employment of neural network models on any raster layer representing a matrix of values or image data. Deep neural networks show a clear improvement in computer vision tasks, enabling the automatic performance of, among others, regression, segmentation and detection of objects in the images. The *Deepness* plugin supports model types that complete the abovementioned tasks, linking deep learning inference directly with the most popular geographic information system (GIS) application. Moreover, a model registry with ready-to-use models is provided, bringing the power of deep learning to users without machine learning expertise. This enables augmenting the familiar, established workflow with new functionalities.

Code metadata

Current code version	0.5.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00338
Permanent link to Reproducible Capsule	–
Legal Code License	Apache License 2.0
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Python 3 on Linux, OSX or Windows. Dependencies listed in setup.py in code repository
If available Link to developer documentation/manual	https://qgis-plugin-deepness.readthedocs.io/
Support email for questions	vision@put.poznan.pl

1. Motivation and significance

Remote sensing is a measurement technique to acquire information about an object without physical contact. Nowadays, it is often associated with imaging methods from satellites, planes, and unmanned aerial vehicles. It provides spatial data information paired with information from Global Navigation Satellite Systems (GNSS), enabling processing and visualisation using a Geographic Information System (GIS). These geotagged images are widely utilised in precise agriculture [1], urban and spatial planning [2,3], and climate change monitoring [4]. What is more, there are a lot of public-available datasets containing geotagged data, for example, ESA's Sentinel [5], NASA Landsat [6], Eurosat land cover [7]. Datasets diversity enables rapid development and growth of a number of applications.

One of the most popular open-source tools for spatial data processing is Quantum GIS (QGIS) [8]. It provides several tools for visualising, processing and analysis, making this software useful for researchers in many fields. Moreover, thanks to its API, it enables expanding functionality through the development of Python plugins. In recent years, many QGIS plugins for image processing have been developed. In [9], a plugin for semi-automatic classification was presented. Their algorithm performs pixel-level classification based on human-annotated labels used for training. Authors of [10] developed a plugin for hyperspectral image analysis accelerated by their domain-specific machine learning models. The authors of [11] developed a QGIS-compatible plugin to demonstrate the potential of their method for neural network training to perform semantic segmentation with human guidance. The plugins leverage the potential created by the use of machine learning for image processing tasks and can be easily incorporated into existing QGIS workflows. However, there is a huge gap in opportunities for the use of deep learning [12] as

* Corresponding author.
E-mail address: marek.kraft@put.poznan.pl (Marek Kraft).

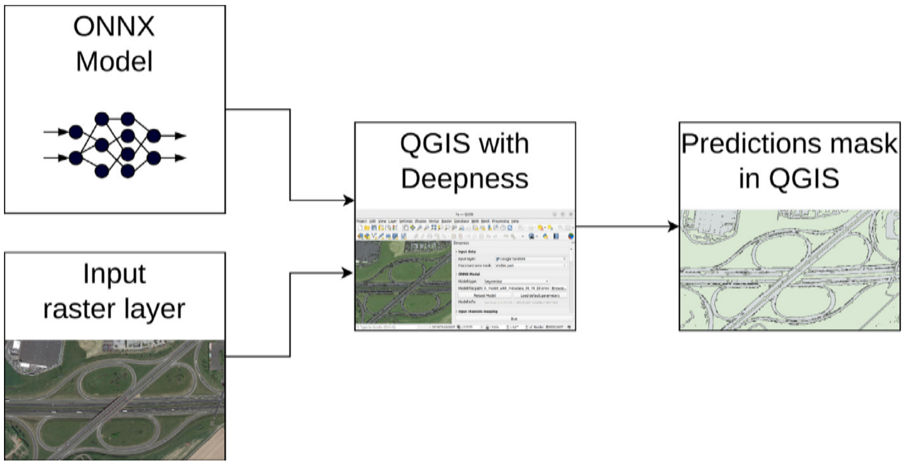


Fig. 1. The schema of *Deepness* operation. The deep learning model is utilised to perform inference on a raster layer, generating results as a new layer in QGIS.

a part of the QGIS workflow. For the tool aimed at non-expert users accustomed to the QGIS workflow and interface, one has to provide a solution that is easy to use and features comprehensive documentation. It is also important to ensure the extensibility of the tool in terms of functions and machine learning models. Therefore, in this paper, we present a *Deepness* – remote-sensing plugin for QGIS that enables data processing using deep-learning models for segmentation, regression and object detection tasks. The main idea of the plugin operation is presented in Fig. 1.

2. Software description

The *Deepness* plugin is easily available to install within the QGIS plugins manager, either downloaded online or installed from a zip file. It connects deep learning expertise with remote sensing applications' needs and is targeted and designed to be operated by three different groups of people:

- **casual users** - final users of the plugin, using the out-of-the-box models in QGIS, either from the model registry or prepared by others. This user group may have no or limited knowledge of machine learning but can benefit from including machine learning-based processing in their workflow. This group includes, e.g. climate researchers, land surveyors, hydrologists, spatial and urban planners, farmers, educators, etc. As such, this group is also potentially much bigger than the crowd of machine learning experts. To minimise the entry threshold, the model usage workflow for this group is simplified to minimum by using readily available models and processing goal-oriented.
- **model creators** - people with machine learning expertise, who might lack domain knowledge. Expected to use the plugin to support model creation by using data export tools and rapid visual verification of the model, encouraged to publish custom models in the model registry.
- **developers** - people with extensive programming knowledge, who want to contribute to the plugin's development, extend the supported model types and architectures, or even reuse its part in custom applications.

2.1. Functionalities of the plugin

The main purpose of the *Deepness* plugin is to run the inference using custom neural network models on any raster layer or combination of raster layers in QGIS. *Deepness* allows the user to employ any model that is stored using framework-agnostic Open Neural Network Exchange (ONNX) format [13], as long as the model's input and output format requirements are met. It is possible thanks to the utilisation of the ONNXRuntime framework [14] in the backend, assuring model compatibility across a range of different combinations of target hardware and software.

There are available three different modes in terms of the area of processing:

- processing of the entire raster layer file,
- processing only area constrained within a polygon, allowing for specifying subregions,
- processing only the part currently visible on the screen, allowing for rapid usage while viewing extensive orthophotos, including huge maps.

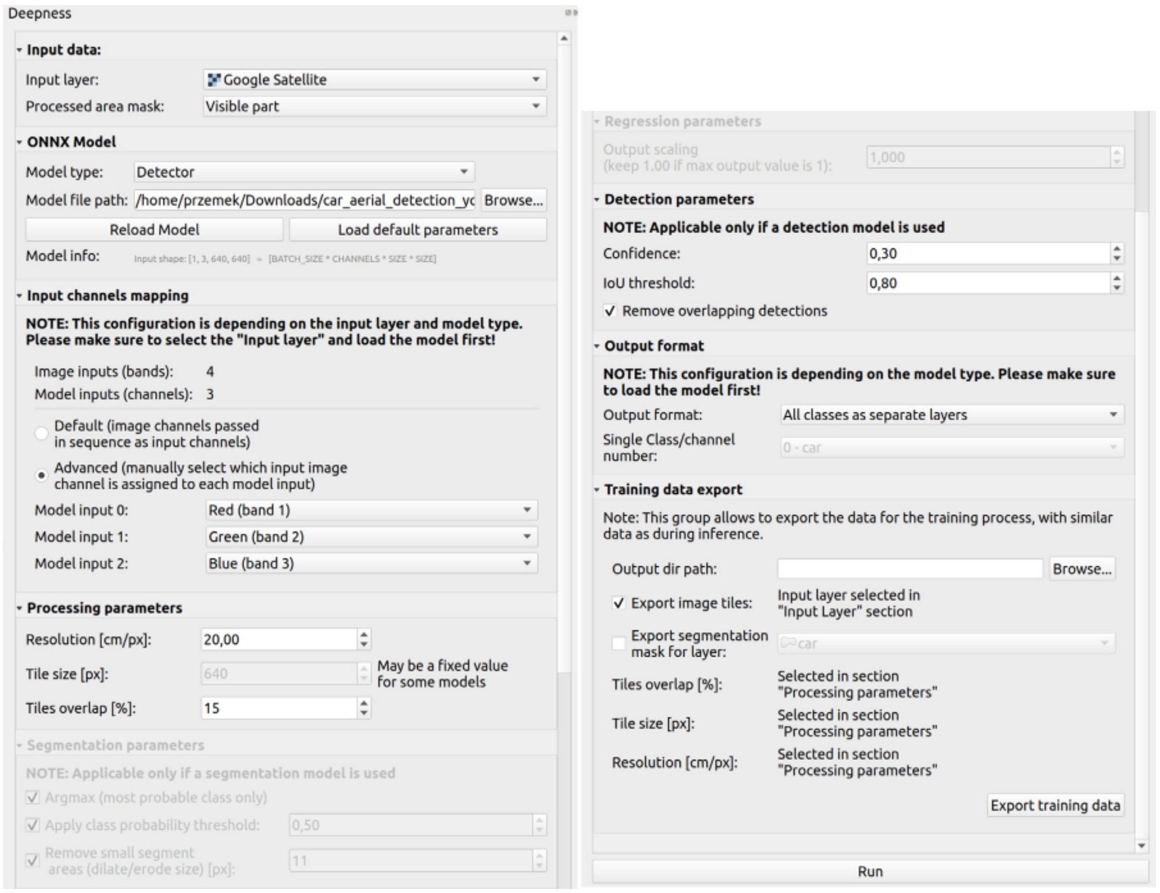
In each of these modes, the processed area size in pixels is usually much bigger than the capability of neural network models. Therefore, the raster layer is divided into small tiles, acceptable by the model, and processed tile by tile, with inference results stitched together.

Deepness is integrated with the QGIS layer system, allowing for processing of any geotagged raster. It includes both custom raster layers, e.g. obtained from UAV mapping, as well as popular maps easily available in QGIS thanks to free tools like QuickMapServices plugin,[1] including Bing Aerial, Google Satellite and Sentinel. The model output data is also saved as a QGIS layer, which maintains all the geospatial information.

Our plugin allows for the three typical types of image-processing tasks performed with the use of neural networks: area segmentation, object detection, and continuous-value regression. The choice of tasks was decided after performing a literature review and based on the number of potential uses for these basic tasks[15,16]. However, with the plugin being a completely open source solution, it can easily be extended to support tasks like e.g. scene classification or descriptor generation [17]. After the processing is finished, a statistics summary report is presented, containing information relevant to the aforementioned task, e.g. in the case of the detection task, the report contains the number of objects of each class that were detected.

To facilitate the machine learning model use for casual users not familiar with this domain, *Deepness* defines a set of model metaparameters, relevant to the model inference and results presentation, which are stored in the ONNX model format. This metadata includes information on preprocessing, like model type, spatial resolution (cm per pixel), names of classes, size of the processed tiles, and tiles' overlap. It also includes mapping of

[1] Available at https://nextgis.com/blog/quickmapservices/



Deepness

Input data:
Input layer: Google Satellite
Processed area mask: Visible part

ONNX Model
Model type: Detector
Model file path: /home/przemek/Downloads/car_aerial_detection_yc Browse...
Reload Model Load default parameters
Model info: Input shape: [1, 3, 640, 640] = [BATCH_SIZE * CHANNELS * SIZE * SIZE]

Input channels mapping
NOTE: This configuration is depending on the input layer and model type. Please make sure to select the "Input layer" and load the model first!
Image inputs (bands): 4
Model inputs (channels): 3
Default (image channels passed in sequence as input channels)
● Advanced (manually select which input image channel is assigned to each model input)
Model input 0: Red (band 1)
Model input 1: Green (band 2)
Model input 2: Blue (band 3)

Processing parameters
Resolution [cm/px]: 20,00
Tile size [px]: 640 May be a fixed value for some models
Tiles overlap [%]: 15

Segmentation parameters
NOTE: Applicable only if a segmentation model is used
✓ Argmax (most probable class only)
✓ Apply class probability threshold: 0,50
✓ Remove small segment areas (dilate/erode size) [px]: 11

Regression parameters
Output scaling (keep 1.00 if max output value is 1): 1,000

Detection parameters
NOTE: Applicable only if a detection model is used
Confidence: 0,30
IoU threshold: 0,80
✓ Remove overlapping detections

Output format
NOTE: This configuration is depending on the model type. Please make sure to load the model first!
Output format: All classes as separate layers
Single Class/channel number: 0 - car

Training data export
Note: This group allows to export the data for the training process, with similar data as during inference.
Output dir path: Browse...
✓ Export image tiles: Input layer selected in "Input Layer" section
Export segmentation mask for layer: car
Tiles overlap [%]: Selected in section "Processing parameters"
Tile size [px]: Selected in section "Processing parameters"
Resolution [cm/px]: Selected in section "Processing parameters"
Export training data

Run

Fig. 2. View of the main *Deepness* plugin UI, allowing to select all essential parameters and run the processing.

image channels to model input channels, allowing to process of not only RGB images but also multispectral data. Missing and incorrect data cases are handled by displaying informative error messages.. The metadata also includes postprocessing information about filtering size, output scaling, class probability threshold (for segmentation) and detection confidence level (for detection). All of these parameters are also exposed in the user interface in the form of selectable widgets, allowing more advanced users, to tune the model behaviour to specific data.

Apart from the core inference-oriented functionality, the plugin allows for exporting raster layer data as small images (called tiles) for training purposes, with parametrisation of spatial resolution, tile size in pixels, overlap and masking layer. This enables the use of QGIS for annotation using its native layer mechanism and for rapid construction of datasets.

Deepness UI (user interface) is shown in Fig. 2.

2.2. Example workflow

Below we demonstrate an example workflow of cars detection with *Deepness* from a casual user perspective:

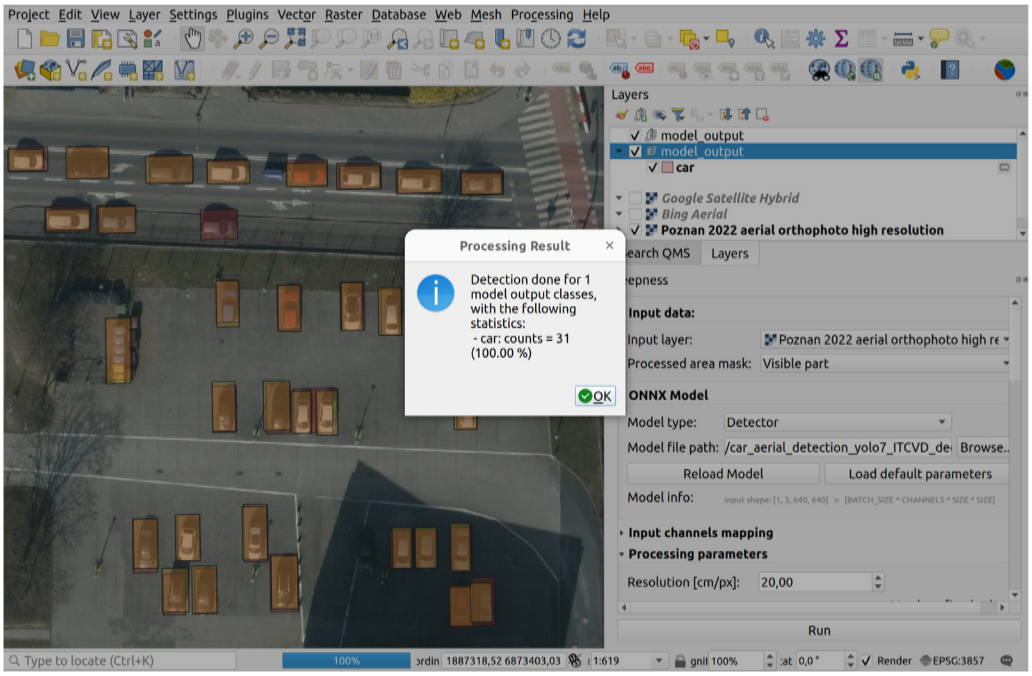
- **Orthophoto loading** - run QGIS and load a custom orthophoto from a file, and zoom in to the area where we want to detect cars.
- **Input data selection** - open the *Deepness* UI panel and in section *Input data* select the right input layers and processing range as *Visible part*.
- **Model selection** - choose a path to the cars detection model file (available in the model registry) and load it. *Deepness* should properly determine and display that it is a detection class task.
- **Model processing parameters selection** - load default model parameters, containing, e.g. information about the spatial resolution on which the model was trained. If needed, modify *Detection parameter* section, e.g. the minimal detection confidence threshold.
- **Running processing** - once all the parameters are set, processing can be started.
- **Results presentation** - after processing is finished, a report with model output is shown, containing information about the number of detected cars. The bounding boxes for detected cars are shown as a new QGIS vector layer and can be saved or exported.

The result of the processing is shown in Fig. 3.

2.3. Supported model types

The plugin supports three common types of deep learning models:

- **Segmentation** - for each input pixel, the network estimate probability of belonging to a predefined class based on its surrounding and global context. One of the most common architectures that for $H \times W \times C$ input image generates $H \times W \times C$ probability output mask is U-Net [18]. Sample applications include buildings segmentation [19] and flood monitoring [20].
- **Regression** - for each input pixel, the network predicts a single continuous value for the regressed variable. For $H \times W \times C$ input image, we expect an output image of resolution $H \times W \times V$, where V is the number of regressed variables, usually equal to 1. Sample applications include predicting


Processing Result

Detection done for 1 model output classes, with the following statistics:
- car: counts = 31 (100.00 %)

Input data:
Input layer: Poznan 2022 aerial orthophoto high re
Processed area mask: Visible part

ONNX Model
Model type: Detector
Model file path: /car_aerial_detection_yolo7_ITCVD_de
Model info: Input shape: [1, 3, 640, 640] = [BATCH_SIZE * CHANNELS * SIZE * SIZE]

Processing parameters
Resolution [cm/px]: 20,00

Fig. 3. Result of an example workflow - cars detection at Poznan University of Technology.

land surface temperature [21], and forecasting vegetation indices [22].

- **Detection** - the most popular object detection methods localise and classify bounding boxes for each object in an image. The recent state-of-the-art real-time object detection network is YOLOv7 [23]. Sample applications include counting cars in a city [24] and detecting weeds in a field [25].

Fig. 4 shows example applications of these three different model types.

Sample models are provided along with the plugin to showcase the use cases of the plugin and demonstrate the process for developing user models for compatibility with the plugin. While there are no technical obstacles to multi-task model implementation, the included example models are single-task neural networks dedicated to either segmentation, regression or object detection.

2.4. Model registry

At the moment of writing the article, six free-to-use models are available, with a summary presented in Table 1. The models have been trained by the authors, utilising the *PyTorch* framework. The training pipelines for car detection and land cover are provided for reference in the repository.

Development of additional models is the focus of planned future work.

All models contain metadata required for their usage in the plugin. The provided models are not universal tools and will perform well only on similar data as in the training datasets. Other researchers are encouraged to add their new models to the registry.

2.5. Implementation details

The model inference is done tile by tile, therefore the entire area of processing needs to be divided into square tiles. Each tile has the same size as the model input size. The division and spatial resampling are done with regard to the overlap and spatial resolution settings. The plugin also handles conversion of the coordinate systems used for the input raster layer, map canvas and restricting polygons, which all can use different coordinate systems. Additionally, the model input channels need to be mapped to the QGIS layer, which itself has different formats of colour channels encoding. After processing each tile by the model, its output is stitched into the final mask or another type of output information, also with regard to the tiles' overlap. Depending on the model type, different type of post-processing is applied. For the segmentation model, small areas of pixels are filtered out before detecting polygons for specific classes. For the detection model, non-maximum suppression and prediction confidence level thresholding are applied (NMS). The plugin is implemented with a modular architecture. Therefore, it is easily extensible to new model classes. The code is written as cross-platform, and it was successfully tested on a few different Windows, Linux and macOS variants.

The preferred device to run the neural network inference is the GPU. If no GPU is found in the system by the *ONNXRuntime* library, it runs on the CPU. With GPU acceleration, inference time is reduced compared to CPU processing. However, the processing time depends on the hardware and software configuration, as well as the model parameters, such as the model architecture and size of the input layer.

3. Impact and applications

Deepness can be downloaded and installed automatically from within the QGIS application, with over 3000 downloads as of 19.05.2023 without relying on active promotion. It has also generated interest in the plugin GitHub repository with a few issues and discussions. Based on the repository stars it is interesting for people around the globe. The authors also received a few email contacts regarding requests for new model functionalities and support. Judging by this information, it seems the users are generally interested not only in the plugin itself but also in the ready-to-use models from the registry. Therefore extending the registry will continue to be an active area of development. Additionally, we noticed that *Deepness* helps spread knowledge

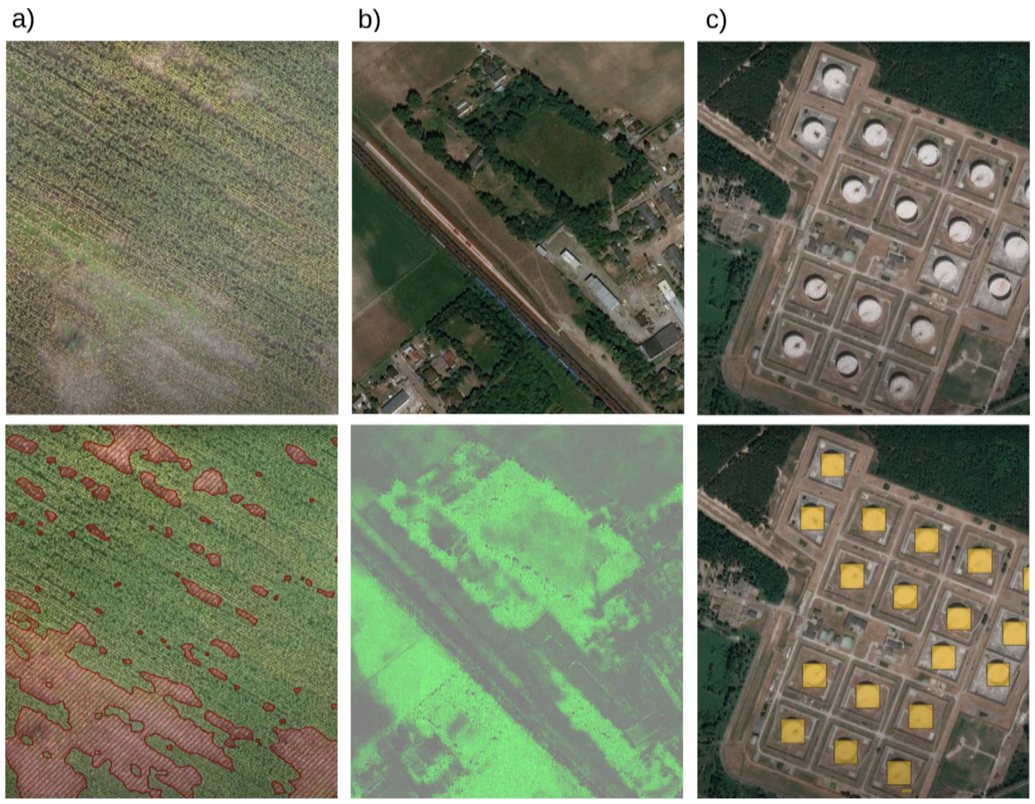


Fig. 4. Example applications of three model types. The upper row is the input orthophoto, the bottom row shows the model output. a) Segmentation model - detected crop damages highlighted in pink hatched lines, on UAV recorded orthophoto; b) regression model - vegetation index calculation. Green saturation corresponds to the vegetation index value; c) detection model - detection of oil storage tanks, marked with yellow rectangles on Bing Maps.

Table 1
Model registry for *Deepness*.

Model functionality	Resolution [cm/pixel]	Description
Land Cover Segmentation	40	The model is trained on the LandCover.ai dataset [26]. It generates masks for the following classes: building, woodland, water, and road. The dataset includes only aerial imagery data collected over Poland.
Corn Field Damage Segmentation	3	Model for Corn Field Damage Segmentation created on own dataset labelled by experts. It generates 3 outputs: healthy crop, damaged crop, and out-of-field area. The dataset is not publicly available.
Roads Segmentation	21	The model segments the Google Earth satellite images into ‘road’ and ‘not-road’ classes. The model works best on wide car roads, crossroads and roundabouts.
Planes Detection	70	YOLOv7 tiny model for plane detection on satellite images. Based on the Airbus Aircraft Detection dataset[a].
Oil Storage Detection	150	YOLOv5-m model for oil tank storage detection on satellite images. Based on the Airbus Oil Storage Detection dataset[b].
Cars Detection	10	YOLOv7-m model for cars detection on aerial images. Based on the ITCVD dataset [24].

[a]Available at https://www.kaggle.com/datasets/airbusgeo/airbus-aircrafts-sample-dataset.
[b]Available at https://www.kaggle.com/datasets/airbusgeo/airbus-oil-storage-detection-dataset.

about machine learning in computer vision, especially about deep learning. It can benefit QGIS users, which often have domain knowledge, but lack experience in machine learning. We hope sharing this knowledge will also inspire them to apply deep learning in their research area to achieve synergistic effect of interdisciplinary research leading to new, exciting results.

4. Conclusions

Deepness plugin already proved to be useful for running inference of neural network models, not only for machine learning engineers but what is even more important for casual users, without machine learning experience. We believe this is to a large extent due to the fact, that it augments the existing workflow, instead of trying to enforce a new, unknown one. Doing this, it truly puts the end users in the front seat, enhancing their productivity. The prepared documentation and examples make it easy to familiarise the users with the tool, its functions and settings. The plugin engine supports the most popular classes of models for segmentation regression and detection, with potential for further improvement.

Models from the registry proved to be especially interesting for the users so far, demonstrating the potential of the tool across sample applications. It has the potential of becoming a popular collection of tools for orthophoto processing in QGIS. Examples of models in the registry, which were trained on available online datasets, allow the detection of cars and planes or to perform land cover segmentation.

Aside from the aforementioned end-user documentation, *Deepness* is also well documented from the perspective of developers and model creators, which enables easy integration of own, customised neural network models within the platform.

Illustrative examples

As an illustrative example, we link a video on youtube presenting our plugin: https://www.youtube.com/watch?v=RCr_ULHHc8A

Funding

This work was supported by the Ministry of Education and Science under Grant 0214/SBAD/0244.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used to prepare the included neural network models is linked in the respective section of the article

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2023.101495.

References

[1] Budiharto W, Chowanda A, Gunawan AAS, Irwansyah E, Suroso JS. A review and progress of research on autonomous drone in agriculture, delivering items and geographical information systems (GIS). In: 2019 2nd World symposium on communication engineering. IEEE; 2019, p. 205–9.
[2] Zaki A, Buchori I, Sejati AW, Liu Y. An object-based image analysis in QGIS for image classification and assessment of coastal spatial planning. Egypt J Remote Sens Space Sci 2022;25(2):349–59. http://dx.doi.org/10.1016/j.ejrs.2022.03.002, URL https://www.sciencedirect.com/science/article/pii/S111098232200031X.
[3] Badach J, Voordeckers D, Nyka L, Van Acker M. A framework for air quality management zones-useful GIS-based tool for urban planning: Case studies in antwerp and gdańsk. Build Environ 2020;174:106743.
[4] Behera M, Sena DR, Mandal U, Kashyap PS, Dash SS. Integrated GIS-based RUSLE approach for quantification of potential soil erosion under future climate change scenarios. Environ Monit Assess 2020;192:1–18.
[5] Drusch M, Del Bello U, Carlier S, Colin O, Fernandez V, Gascon F, et al. Sentinel-2: ESA's optical high-resolution mission for GMES operational services. Remote Sens Environ 2012;120:25–36.
[6] Tucker CJ, Grant DM, Dykstra JD. NASA's global orthorectified landsat data set. Photogramm Eng Remote Sens 2004;70(3):313–22.
[7] Helber P, Bischke B, Dengel A, Borth D. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. IEEE J Sel Top Appl Earth Obs Remote Sens 2019;12(7):2217–26.
[8] QGI Development Team. QGIS geographic information system. QGIS Association; 2023, URL https://www.qgis.org.
[9] Congedo L. Semi-automatic classification plugin: A python tool for the download and processing of remote sensing images in QGIS. J Open Source Softw 2021;6(64):3172.
[10] Lyngdoh RB, Sahadevan AS, Ahmad T, Rathore PS, Mishra M, Gupta PK, et al. AVHYAS: A free and open source QGIS plugin for advanced hyperspectral image analysis. In: 2021 International conference on emerging techniques in computational intelligence. IEEE; 2021, p. 71–6.
[11] Lenczner G, Le Saux B, Luminari N, Chan-Hon-Tong A, Le Besnerais G. DISIR: Deep image segmentation with interactive refinement. ISPRS Ann Photogramm, Remote Sens Spatial Inf Sci 2020;2:877–84.
[12] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436–44.
[13] Bai J, Lu F, Zhang K, et al. ONNX: Open neural network exchange. 2019, https://github.com/onnx/onnx,
[14] developers OR. ONNX runtime. 2021, https://onnxruntime.ai/, Version: 1.12.
[15] Hoeser T, Bachofer F, Kuenzer C. Object detection and image segmentation with deep learning on earth observation data: A review—Part II: Applications. Remote Sens 2020;12(18):3053.
[16] Ma L, Liu Y, Zhang X, Ye Y, Yin G, Johnson BA. Deep learning in remote sensing applications: A meta-analysis and review. ISPRS J Photogramm Remote Sens 2019;152:166–77.
[17] Cheng G, Xie X, Han J, Guo L, Xia G-S. Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities. IEEE J Sel Top Appl Earth Obs Remote Sens 2020;13:3735–56.
[18] Minaee S, Boykov Y, Porikli F, Plaza A, Kehtarnavaz N, Terzopoulos D. Image segmentation using deep learning: A survey. IEEE Trans Pattern Anal Mach Intell 2021;44(7):3523–42.
[19] Rafique MU, Jacobs N. Weakly supervised building segmentation from aerial images. In: IGARSS 2019-2019 IEEE international geoscience and remote sensing symposium. IEEE; 2019, p. 3955–8.
[20] Muhadi NA, Abdullah AF, Bejo SK, Mahadi MR, Mijic A. Image segmentation methods for flood monitoring system. Water 2020;12(6):1825.
[21] Jia H, Yang D, Deng W, Wei Q, Jiang W. Predicting land surface temperature with geographically weighed regression and deep learning. Wiley Interdiscip Rev: Data Min Knowl Discov 2021;11(1):e1396.
[22] Ferchichi A, Abbes AB, Barra V, Farah IR. Forecasting vegetation indices from spatio-temporal remotely sensed data using deep learning-based approaches: A systematic literature review. Ecol Inform 2022;101552.
[23] Zou Z, Chen K, Shi Z, Guo Y, Ye J. Object detection in 20 years: A survey. Proc IEEE 2023.
[24] Yang MY, Liao W, Li X, Rosenhahn B. Deep learning for vehicle detection in aerial images. In: 2018 25th IEEE international conference on image processing. IEEE; 2018, p. 3079–83.
[25] Peng H, Li Z, Zhou Z, Shao Y. Weed detection in paddy field using an improved RetinaNet network. Comput Electron Agric 2022;199:107179.
[26] Boguszewski A, Batorski D, Ziemba-Jankowska N, Dziedzic T, Zambrzycka A. LandCover.ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR) workshops. 2021, p. 1102–10.