# Performance evaluation and limitations assessment of GeoAI democratization for natural hazard induced disasters

# Performance evaluation and limitations assessment of GeoAI democratization for natural hazard induced disasters

Andrea Demartis[1], Fabio Giulio Tonolo[1], Andrea Ajmar[2]

1 Politecnico di Torino, Department of Architecture and Design (DAD) – LabG4CH, Viale Mattioli 39, 10125, Torino, Italy
2 Politecnico and Università di Torino, Interuniversity Department of Regional and Urban Studies and Planning (DIST), Viale Mattioli 39, 10125, Torino, Italy

andrea.demartis@polito.it, fabio.giuliotonolo@polito.it, andrea.ajmar@polito.it

**Keywords:** GeoAI, Deep Learning, EO, Foundation Models, Floods, Wildfires.

**Abstract**

Recent advances in Deep Learning (DL) for Earth Observation (EO) have enabled the deployment of increasingly complex pre-trained models within operational geospatial workflows. Among these, EO foundation models offer promising opportunities for hazard mapping from multispectral satellite data, while interfaces integrated in GIS software aim to make such models accessible also to users with limited coding expertise. In this context, this study investigates the use of pre-trained models for burn scar and flood extent segmentation from Sentinel-2 imagery based on Prithvi, comparing their performances in a commercial GIS software and in a standalone Python environment using common datasets. The analysis focuses on the reproducibility of the outputs across different environments, as well as on the trade-off between accessibility, transparency, and methodological control. Results show that the GIS based platform and the reconstruction in Python were generally aligned, confirming the validity of the interface. Agreement was stronger for burn scar segmentation, whereas flood extent mapping showed greater variability, likely due to the intrinsically more complex characteristics of flooded areas which can yield different results even with small differences in model building. An additional exploratory experiment was conducted using the prompt-based model CLIPSeg for cloud masking in the flood workflow. Although its contribution proved limited and strongly scene-dependent, it provided useful insight into the role of auxiliary prompt-based models within EO pipelines. Overall, the study shows that GIS-based DL interfaces effectively democratize access to model deployment, while coding-based environments remain essential for transparency, customization, and broader methodological control.

## 1. Introduction

Natural Hazard Induced Disasters (NHID) such as floods and wildfires are currently a serious problem. Several studies show the severity and frequency of these phenomena are increasing(Wang *et al.*, 2022; Cunningham, Williamson and Bowman, 2024). The impact of these events requires tools for monitoring, emergency support, and post-event damage estimation. In this context Earth Observation (EO) has been playing a central role. Thanks to the spatial coverage, temporal, spectral and spatial resolution provided by satellite platforms, the discipline is able to provide support in all stages of a NHID management (Le Cozannet *et al.*, 2020). Specifically, for open multispectral satellite data, the Copernicus Sentinel-2 mission assumed a prominent role. Its temporal resolution and global coverage allow for monitoring, retrieving pre-event conditions, and post-event affected areas delineation workflows (Poursanidis and Chrysoulakis, 2017). However, especially when based on manual labelling and interpretation, their use in rapid mapping contexts remains time consuming. For this reason, the research for faster tools to exploit these data remains active at the present day.

Deep Learning (DL) has made steps forward in this direction, emerging as a promising tool to shorten inference times, for automation, and other application such as pattern recognition (Ma *et al.*, 2019). Among the latest advancements in the field of EO with DL it is possible to find Foundation Models (FM) like PrithviEO, namely Artificial Intelligence (AI) systems trained on vast and diverse geospatial data that can be adapted to specific tasks with fine-tuning (Huo *et al.*, 2025). In this context, these models are usually trained with datasets of multispectral imagery, that allow the model to exploit the different relationship between bands. Another recent advancement in DL is represented by multimodal Prompt-Based Models (PBMs), namely detection models paired with a text-encoder that allow the detection of any object through zero-shot-classification (Lüddecke and Ecker, 2021). Their deployment, however, requires non-trivial coding abilities, which might be a barrier for most users (Hoeser and Kuenzer, 2020).

To address this gap, in both open and commercial Geographic Information Systems (GIS) software Graphic User Interface (GUI) based interfaces have been recently introduced (Aszkowski *et al.*, 2023) (Choi, 2023). These platforms allow users to run pre-trained models, fine-tune existing models, and even train models from scratch. These interfaces effectively lower the initial technical barrier, shifting the expertise from coding and DL knowledge to results assessment, democratizing access to the tool to more users. Nevertheless, while this democratization lowers the accessibility barrier to users and allows rapid experimentation and visualization, it necessarily lowers customization capabilities and transparency of the model. Constraints might be limited hyperparameters availability during the inference step, hidden pre-processing or post-processing in the pre-trained model. On the contrary, coding environments remain more flexible, allowing customization in each step of the DL workflow. The practical relevance of this trade-off is that lower customization might yield unprecise output. A specific field in which this problem might be relevant is the rapid mapping workflow, where

methodological transparency with operational and predictive efficiency can directly affect the usefulness of the results.

Within this specific context, this study investigates the use of pre-trained DL models for flood and burn scar segmentation available in a commercial GIS software using datasets of Sentinel-2 imagery assessing the thematic accuracy with respect to ground truth dataset. It then compares the GUI based workflow with an equivalent coding-based implementation in order to assess the comparability of the results in terms of performance, degree of control, and inference time. In addition, the study includes an exploratory evaluation of PBM for cloud segmentation, examining whether their integration can improve the final result. Overall, the paper aims at providing a analytical evaluation of democratization of DL in operational EO workflows.

## 2. Materials and Methods

### 2.1 Study Design

The study was implemented in two different environments: the commercial GIS software ArcGIS Pro and a standalone Python environment. ArcGIS Pro was used to run the pre-trained models available in the Living Atlas (LA), ESRI’s database where it’s possible to find pre-trained DL models compatible with the tools in the software. Through reverse engineering the same models were reconstructed outside the GIS software, to assess the reproducibility of outputs and customization. Since the focus of the research is on comparability and not on the best possible result in terms of thematic accuracy, all models were inferenced using default hyperparameters. The research was carried out in ArcGIS Pro 3.6.1, while the standalone implementation was developed in Python 3.10.20 using PyTorch (Paszke *et al.*, 2019) and MMSegmentation (MMSegmentation Contributors, 2020). The workstation used is equipped with an Intel Core i9-14900KF CPU @ 3.200 Mhz, 64 GB RAM, an NVIDIA GeForce RTX 4070 GPU with 12 GB VRAM, and Windows 11 Pro.

### 2.2 Datasets

**2.2.1 Wildfire Dataset:** For the wildfire branch of the research, a dataset composed of activations from the Copernicus Emergency Mapping Service (CEMS) was selected (Arnaudo *et al.*, 2023). The dataset is composed of 171 wildfire events images subdivided per activation and area of interest. Images are stored as stacked *.tif* of the full product of the Sentinel-2 service, composed of 12 multispectral bands at different resolutions. Paired with each Sentinel-2 image a mask of the burn scar is available, along with Land Cover (LC) maps from different sources, such as ESA, Esri and Copernicus Land Monitoring Service. For each *.tif* file a *.png* version is available to ease visualization. To balance the analysis with the flood branch, only the test split of the dataset was used, resulting in 87 images.

**2.2.2 Flood Dataset:** For the flood branch of the research, the selected dataset is *WorldFloods v2* (Portalés-Julià *et al.*, 2023), a dataset of 509 pairs of Sentinel-2 images and flood segmentation masks coming from different sources, usually optical and radar satellites. The dataset is composed by: the Sentinel-2 images, stored as a 15 bands stack of the Sentinel-2 data; a Permanent Water (PW) raster layer coming from the Joint Research Center (JRC) subdivided according to the seasonality of the PW; Ground Truth (GT) raster masks composed of 2 bands, one for cloud masking and one for flood masking. It is important to note that in the flood masks the full extent of PW is included, therefore being actually the observed water extent rather than the flooded areas. This difference highlights the importance of a standard terminology in the emergency management domain, to avoid misleading the end users (International Working Group on Satellite-based Emergency Mapping (IWG-SEM), 2015).
Given the rapid temporal dynamics of flood events, particular care was taken to select only those image–label pairs for which the time difference between the Sentinel-2 acquisitions and the data used to derive the flood ground-truth mask was lower than 1 hour. This subset of the dataset, resulted in 61 images, was then used to test the model.

### 2.3 Models

Recent advancements in DL for EO have led to the emergence of FMs, namely models that are pre-trained on a large and diverse dataset to learn the general feature representations and spectral relationship between bands. The main advantage of FMs lies in the possibility of fine-tuning for a variety of task specific with the same pre-trained version, reducing the dependence on large task-specific labelled datasets. Examples of FMs include *Terramind* (Jakubik *et al.*, 2025) and, like in the case of this research, in *Prithvi EO* (Szwarcman *et al.*, 2024).

Alongside FMs, increasing attention has been given to PBMs, namely models in which during inference the model is guided by a user defined prompt of the target, usually in textual form. They support flexible target extraction, including zero-shot applications. From an operative perspective, they are particularly useful in contexts where classes are not defined a priori or class specific models are not available (Demartis *et al.*, 2026).

**2.3.1 Prithvi - Flood Segmentation:** For flood extent mapping, an available pre-trained model derived from the Prithvi EO FM developed by NASA and IBM by fine-tuning Prithvi-100M on Sen1Floods11 was used (IBM NASA Geospatial, 2023). The model is distributed in the LA as a *.dlpk* file for direct deployment through the DL interface of the ArcGIS Pro software. It was possible to open this file as a compressed folder in order to access the trained checkpoint (*.pth*), the model configuration file (*.py*), the Esri model definition (*.emd*), and the Python wrapper used by the software during inference, all fundamental information needed for the reconstruction of the model in a Python environment. The model works on 6 Sentinel-2 bands, namely Bands 2, 3, 4, 8A, 11, and 12, corresponding to Blue (B), Green (G), Red (R), narrow Near-InfraRed (NIR), and the two Short-Wave InfraRed (SWIR, 1 and 2) bands, that must be provided in this exact order. According to the model configuration file, the input image is subdivided in patches of 224x224 pixels and the output consists of two classes, *No-Water* and *Water/Flood*. It is worth noting how the documentation in the LA refers to a 3 classes output raster, with a third class *NoData/Clouds*; this class, however, cannot be found in the model configuration file. Lastly, input patches are scaled and normalized according to parameters learnt by the model during training and defined in the model configuration file. The model is supposed to work better with Harmonized satellite data (Claverie *et al.*, 2018), but is supposed to work also with level-2 products of Sentinel-2 as reported in the LA documentation of the model. This model was executed directly in ArcGIS Pro through the dedicated DL interface and reconstructed as closely as possible in Python to reproduce the same inference behaviour.

**2.3.2 Prithvi – Burn Scar Segmentation**: For the burn scar extension segmentation, a second pre-trained model based on the Prithvi EO FM was used (IBM NASA Geospatial, 2023). Similarly to the flood segmentation model, the model is distributed through the LA as a *.dlpk* file containing the same files previously mentioned. The model works on the same six Sentinel-2 bands used by the flood segmentation model, namely Bands 2, 3, 4, 8A, 11, 12. Inferencing logic in terms of model construction and pre-processing was therefore the same of the flood segmentation model. Analogously, this model was executed directly in ArcGIS Pro through the dedicated DL interface and reconstructed as closely as possible in Python to reproduce the same inference behaviour.

**2.3.3 Cloud filtering in flood extent segmentation:** For cloud masking, two different models were selected according to what the environment allowed; specifically, in the ArcGIS Pro workflow a DL model available in the LA was selected, while for the standalone Python workflow different repositories were explored.

For the ArcGIS Pro workflow a semantic segmentation model based on CLIPSeg was used (Lüddecke and Ecker, 2021). Unlike the other models used in this research, it is not trained exclusively for a specific subset of objects, but it belongs to the family of PBMs in which the target is specified during inference in textual form, which in this case was “clouds”. Differently from the other models, the required input is an 8-bit RGB image, which was appositely generated with the original Sentinel-2 bands. The resulting output is a binary raster mask of the target object (clouds in this case), which was evaluated using the cloud mask available in the GT of the *WorldFloods* dataset.

For the Python standalone workflow, the OmniCloudMask DL model was selected (Wright *et al.*, 2025). The model is trained to work effectively with geometric resolutions ranging from 10 m to 50 m using R-G-NIR imagery, that was reconstructed from the Sentinel-2 data available in the dataset. The output masks are divided in dense clouds, thin clouds, cloud shadows (that were not used in this research), and non-covered pixels. Subsequently, the obtained masks were evaluated against the GT of the *WorldFloods* dataset.

### 2.4 Preprocessing

For both burn scar and flood extent segmentation, input data was prepared identifying and extracting single bands from the original Sentinel-2 images available in the dataset. Said bands, previously described, were then combined together in a composite raster respecting the required order. A similar preprocessing workflow was followed to build the input of CLIPSeg, where RGB composite rasters stored as 8bit integers were created and then used as input for the PBM.

### 2.5 Inferencing Workflow

The tests were subdivided mainly in two branches, namely inferencing the models in ArcGIS Pro and inferencing the reconstructions in a Python standalone environment.

**2.5.1 Inferencing workflow in ArcGIS Pro**: For both burn scar and flood extent segmentation inference was performed with the “*Classify Pixels Using Deep Learning*”. The GUI allows for inference on a single input raster and batch inference on more input rasters; it also provides an equivalent Python script that calls the tool in a Jupyter Notebook, a coding environment integrated inside ArcGIS Pro. This last option was followed for batch inference on the subsets of datasets. It is important to highlight that launching the tool using GUIs and calling it in Python yields the exact same result, and is not equivalent to recreating the model in a Python environment and inferencing without the use of the ArcGIS Pro tool. The same Python command was used in ArcGIS Pro to inference CLIPSeg for cloud mask segmentation.

**2.5.2 Inferencing workflow in Python**: For both burn scar and flood extent segmentation the models were reconstructed as closely as possible to the models available in Esri LA. To achieve the closest possible result to ArcGIS Pro workflow, the reconstruction was based on the files contained in the original *.dlpk*, namely the model configuration file, the Esri model definition, the Python wrapper, and the trained weights. This allowed for the reconstruction of the models in a standalone Python environment using *PyTorch* and *MMSegmentation*. The resulting models use the same pre-processing strategy and inference hyperparameters identified in the GIS software. Specifically, inference on 224 x 224 pixels patches, with padding= 56 pixels, stride= 112 pixels, batch_size= 4. Lastly, stitching between patches was performed using Hann-like weights; while it was possible to understand that also the ArcGIS Pro model uses this approach, the precise weights were not explicit and empirically deduced. The outputs were stored as raster predictions as the outputs generated in ArcGIS Pro, and evaluated with the same procedure. It is important to highlight that while the reconstruction aim was to emulate ArcGIS Pro’s behaviour, the Python standalone version allows for the change of any pre-processing or inferencing parameter.

### 2.6 Cloud Masking Segmentation workflow

Especially in the flood segmentation workflow, cloud-covered areas represented a relevant source of false positives. For this reason, a cloud-filtering strategy was introduced to improve both predictions and evaluations. A different strategy according to what the limitations allowed was implemented for each workflow.

In the ArcGIS Pro workflow, although task-specific cloud masking models were available in the LA, the input data required was not compatible with the imagery of the dataset; therefore, other options were explored. While not being task-specific and supposedly less performing, CLIPSeg was one of the few DL-based options in the LA for cloud masking compatible with the imagery available in the dataset.

In the Python standalone workflow, given the input data available in the dataset, the task specific OmniCloudMask model was selected.

## 3. Results and Discussion

After inferencing the models in each workflow and evaluating the predictions with the GTs available in the datasets, the results of the models were compared according to the NHID analysed.

### 3.1 Wildfire segmentation: ArcGIS Pro vs Python

The comparison between the Prithvi-based burn scar segmentation model executed in ArcGIS Pro and in Python produced comparable results. In more than 80% of the scenes the absolute differences in F1 scores were lower than 0.1, with a maximum difference of 0.36. An example of close agreement between the output masks is reported in Figure 1, while the highest difference scene is reported in Figure 2. In Figure 2, especially in the ArcGIS Pro prediction (second row, second column), is also possible to observe a recurring phenomenon encountered in both workflows and regardless of the model type, which is the interruption of detection with sharp edges; while it is not possible to understand completely the causes of those artifacts, it is reasonable to think that they are associated with the patching and stitching pre-processing of the original imagery. Between the two approaches, the average absolute differences in terms of F1 score is 0.03 with a standard deviation of 0.07, confirming that the reconstruction in Python is comparable with the model available in ArcGIS Pro. Moreover, is interesting to highlight that the Python reconstruction of the model was able to yield better results than the available version in ArcGIS Pro in terms of F1 Score. Lastly, the model returned an average F1 score of 0.75 both in the Python and ArcGIS Pro workflow.

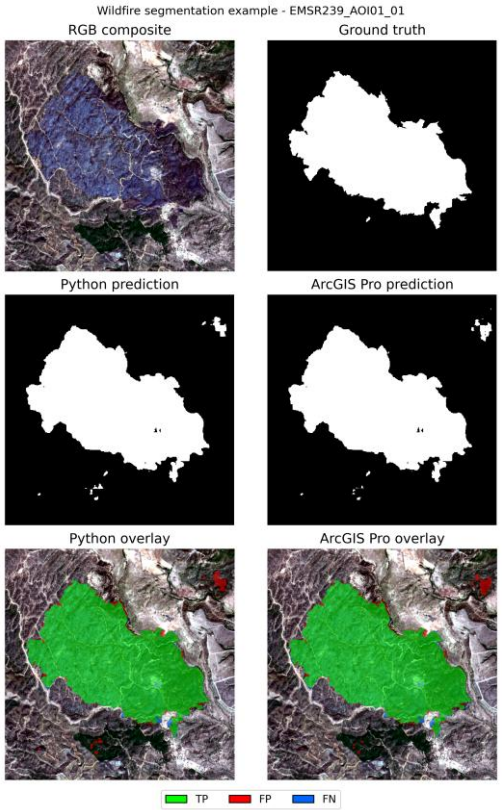


**Figure 1** Comparison between outputs of burn scar segmentation coming from Python and ArcGIS Pro workflow with no F1 score variation

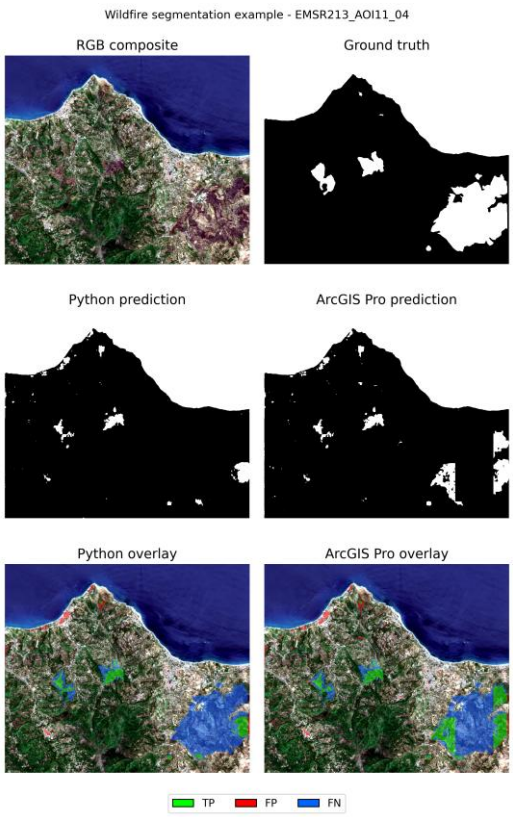


**Figure 2** Comparison between outputs of burn scar segmentation coming from Python and ArcGIS Pro workflow with significative differences (∆F1 Score = 0.23)

### 3.2 Flood segmentation: ArcGIS Pro vs Python

The comparison between the Prithvi-based flood extent segmentation model executed in ArcGIS Pro and in Python produced slightly less comparable results, but still more than 70% of the scenes had a mean absolute difference in terms of F1 score lower than 0.1. Figure 3 shows an example of close agreement between the two workflows, where is also clear how impactful the clouds are, underlining why operational workflows mask clouds beforehand; Figure 4 shows an example where the two models show larger discrepancies. Between the two approaches the mean of absolute differences in terms of F1 score is 0.06, which makes them comparable and consistent with the wildfire reconstruction results.

In light of the results, the flood extent segmentation model is less consistent than the model fine-tuned on burn scars segmentation in terms of F1 Score. This most likely happens due to the nature of the flood phenomenon itself. The model is trained to detect water and relies on multispectral information; flooded water, however, is not pure water but is usually mixed with mud and vegetation, which strongly change the spectral response of the water. Moreover, cloud coverage furtherly increases the number of false positives.

The average F1 score of the models is 0.51 and 0.55 for the ArcGIS Model and its reconstruction in Python respectively. Since cloud coverage was heavily affecting the model’s output, a dedicated cloud-filtering strategy was further investigated.

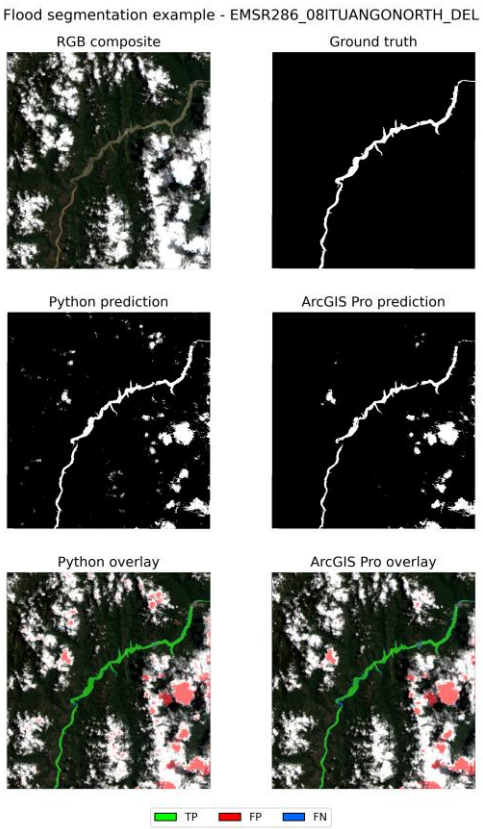


**Figure 3** Comparison between outputs of flood segmentation from Python and ArcGIS Pro workflow (no F1 Score variation)

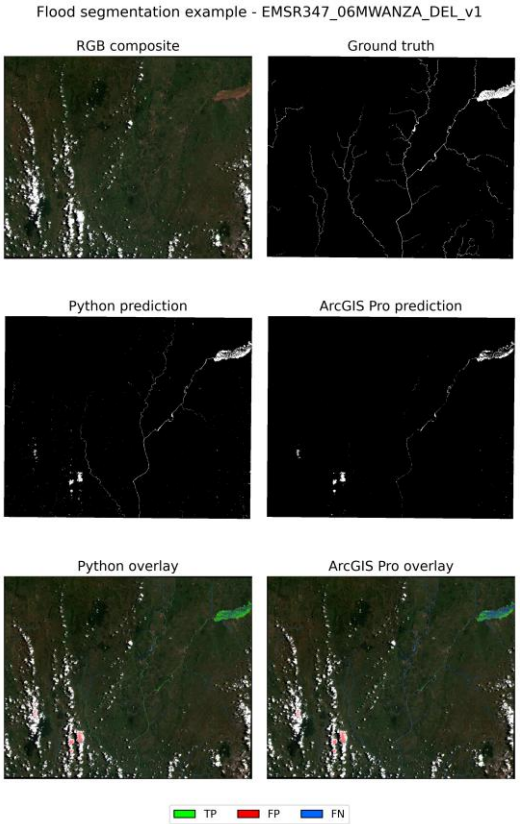


**Figure 4** Comparison between outputs of flood segmentation coming from Python and ArcGIS Pro workflow with significant differences (△F1 Score = 0.22)

### 3.3 Cloud Filtering with CLIPSeg and OmniCloudMask

Although CLIPSeg was able to identify cloudy areas in some scenes, it scored an F1 score higher than 0.5 only in 30% of scenes, and 0.38 on average. Nevertheless, the masks were used as an auxiliary filter, since it was considered unlikely that false positives in cloud segmentation corresponded to flooded areas. While this assumption proved to be reasonable, the impact of CLIPSeg in cloud segmentation was significative only in 7 scenes out of 61, where its usage provided an improvement higher than 0.1 in terms of F1 Score. It is important to highlight, however, how in 2 scenes the results after masking worsened of 0.05 in terms of F1 score. These results show that CLIPSeg cannot be used for systematic cloud filtering, but could be helpful according to scene. Figure 5 shows an example of cloud masking from CLIPSeg. The model seems to be able to recognize only the denser portion of clouds, while in the GT mask also the lighter portion, generally referred to as haze. While haze can impact the radiometry, especially in the RGB bands, its impact in infrared bands is limited and therefore the GT mask used in this research is arguably too precautious, and led to a biased evaluation of the results of CLIPSeg. This also highlight how the cloud masking problem is subjected to interpretation of the user, since the precise delineation might differ. Figure 6 shows an example of how in infrared bands is possible to extract information also through haze. Its role in the workflow should therefore be interpreted as exploratory and supportive rather than as a definitive cloud-masking solution.

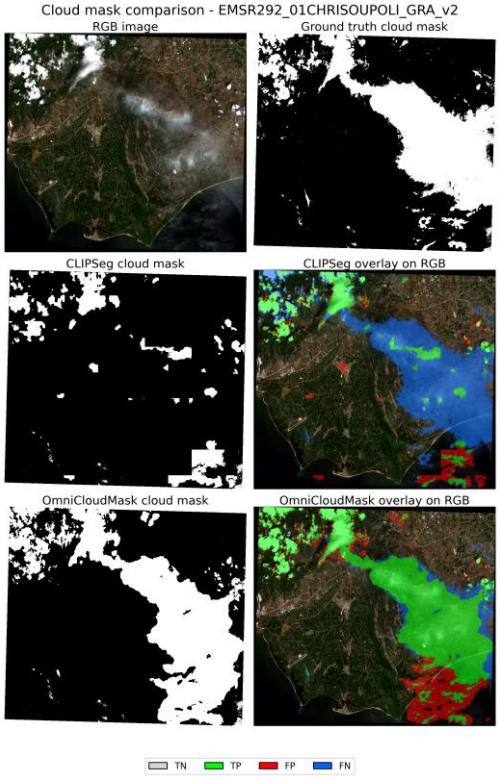


**Figure 5** Example of cloud masking using CLIPSeg and OmniCloudMask (thin and dense clouds)

Concerning OmniCloudMask, the average F1 Score returned by the model is 0.51, highlighting better agreement with the GT masks available in the dataset despite their precautious nature. Cloud masking with OmniCloudMask produced higher maximum improvements, up to 0.48 in terms of F1 Score, and 24 scenes had an improvement higher than 0.1. Despite from a qualitative point of view, the masks produced by OmniCloudMask seems more aligned to what an expert user would label as cloud, the mean improvement in terms of F1 results to be equal to 0.07, furtherly suggesting that further inspection of the GT masks is required for evaluation purposes of the models.

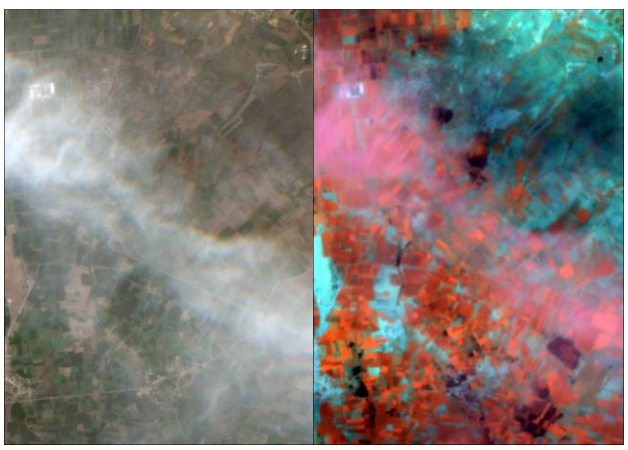

**Figure 6** Example of haze heavily affecting true color (composite RGB, on the left) and limitedly affecting infrared bands (composite NIR-SWIR1-SWIR2, on the right)

### 3.4 Democratization Assessment

The results obtained in this study show that the dedicated DL platform in ArcGIS Pro effectively allow the integration of the supported pre-trained DL models, effectively contributing to the democratization of the tool. In both branches of the study, ArcGIS Pro allowed the deployment of complex DL models through a simplified GUI- based environment, that would have otherwise required non trivial integration of environment management and coding skills. Since the ArcGIS Pro outputs were comparable with the results obtained from the standalone Python reconstruction, it is suggested that, from an operative perspective these interfaces can already provide meaningful results to users with limited programming expertise.
At the same time, it is also highlighted how transparency remains partial. Although it was possible to reconstruct the model through inspection and reconstruction of the available *.dlpk*s, some components remained embedded in the software. Moreover, the reconstruction highlighted how by inferencing the model in a standalone Python environment the user has a much higher degree of freedom in customization, meaning that GUI based DL interfaces lower the initial technical barrier, but do not eliminate the need for coding-based environments when optimization or debugging is needed. In addition, from a computation perspective the standalone Python workflow proved to be more efficient, with bulk inference times approximately three times lower compared to the bulk integration in ArcGIS Pro; while a specific cause for higher computational times cannot be precisely delimited, it is reasonable to think that the integration with proprietary tools of the GIS software slows the computational capabilities, while in a standalone Python environment no other resources must be allocated.

Moreover, inferencing the models in a GIS software provide a major practical advantage in terms of visualization. The possibility to inspect and integrate outputs in a georeferenced workspace made it possible to rapidly understand spatial context and rapidly compare them without the need to proper code an adequate visualizer, which is necessary in a standalone Python environment.

Lastly, it is important to highlight how in the current implementation it is not possible to integrate any model into these interfaces, but only the models that were adapted into a *.dlpk* in the LA; this can limit options and applications. As shown by the cloud segmentation task of this paper, since the only available Cloud Segmentation model was not compatible with the imagery available in the dataset, it was not possible to use another task specific model. In a Python standalone environment, it would have been possible to test other options that require input data compatible with the imagery, potentially obtaining better results.

Therefore, the current democratization of GeoAI in commercial software mainly allows new user to test and use DL models selected from the available ones rather than ensuring full methodological control.

### 3.5 Limitations

This study presents some limitations that should be considered when interpreting the results. First, the reconstruction of the models in the Python standalone environment was designed to reproduce the ArcGIS Pro implementation as closely as possible. This choice was taken to ensure reproducibility of the results obtained in the software, and to improve the results

obtainable with the reconstruction. Future work may explore optimization of the model to improve thematic accuracy of the results, especially exploring the parameters that are not usually available in GUI interfaces. Moreover, although the reconstruction was based on the same model packages files, the replica is not a certified copy of the model, and residual disagreement between the environments may be due to implementation details that were embedded in the GIS software and not fully accessible.

Secondly, CLIPSeg cloud masking should be considered as an exploratory experiment. PBM, being not specifically trained for any specific task, needs to be tested for the specific target object of the analysis. The results of the experiment showed that the model underperformance is given by the not clear geometric boundary, as well as semantic definition of low-densitiy clouds or haze, producing false negatives.

Lastly, the models analysed in this study belong to a rapidly evolving field, which means that new releases of available DL models (or completely new models) in the LA are frequently updated; therefore, with more recent releases results might change.

## 4. Conclusions

This study investigated the use of pre-trained DL in the dedicated interface in the commercial GIS software ArcGIS Pro, comparing whether their results were reproducible in a Python environment, and how the usage of a GUI-based environment limits the freedom of the user. The results show that the Python reconstruction of the model is consistent with the ArcGIS Pro version of the model. The agreement of the models was stronger in burn scar segmentation compared to flood segmentation, which is probably because the phenomenon is intrinsically more ambiguous, so even small differences in model construction can lead to different results.

The additional cloud masking experiment with CLIPSeg showed that PBMs can provide auxiliary support within a workflow, but their contribution remains still scene and target object dependent. In particular, its usage for cloud masking produced systematic improvements in flood segmentation only in a limited number of scenes, probably because delineating cloud-cover boundaries and distinguishing them from haze is inherently difficult, even for an experienced image analyst.

Lastly, from a democratization point of view, the study confirms that the interfaces effectively lower the technical barrier required to run EO DL models, allowing the deployment of complex architectures through user friendly interfaces. However, their integration greatly lowers the transparency of both the model architecture and inferencing possibilities, with pre-processing embedded in the model (e.g. patching and stitching) not clearly explained. Moreover, the democratization is currently limited to the models available in the supported format in the LA, that furtherly limit the options available to a user not able to adapt the target model to the specific format.

## References

Arnaudo, E. *et al.* (2023) “Robust Burned Area Delineation through Multitask Learning.” arXiv. Available at: https://doi.org/10.48550/ARXIV.2309.08368.

Aszkowski, P. *et al.* (2023) “Deepness: Deep neural remote sensing plugin for QGIS,” *SoftwareX*, 23, p. 101495. Available at: https://doi.org/10.1016/j.softx.2023.101495.

Choi, Y. (2023) “GeoAI: Integration of Artificial Intelligence, Machine Learning, and Deep Learning with GIS,” *Applied Sciences*, 13(6), p. 3895. Available at: https://doi.org/10.3390/app13063895.

Claverie, M. *et al.* (2018) “The Harmonized Landsat and Sentinel-2 surface reflectance data set,” *Remote Sensing of Environment*, 219, pp. 145–161. Available at: https://doi.org/10.1016/j.rse.2018.09.002.

Cunningham, C.X., Williamson, G.J. and Bowman, D.M.J.S. (2024) “Increasing frequency and intensity of the most extreme wildfires on Earth,” *Nature Ecology & Evolution*, 8(8), pp. 1420–1425. Available at: https://doi.org/10.1038/s41559-024-02452-2.

Demartis, A. *et al.* (2026) “Analytical Assessment of Pre-Trained Prompt-Based Multimodal Deep Learning Models for UAV-Based Object Detection Supporting Environmental Crimes Monitoring,” *Geomatics*, 6(1), p. 14. Available at: https://doi.org/10.3390/geomatics6010014.

Hoeser, T. and Kuenzer, C. (2020) “Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends,” *Remote Sensing*, 12(10), p. 1667. Available at: https://doi.org/10.3390/rs12101667.

Huo, C. *et al.* (2025) “When Remote Sensing Meets Foundation Model: A Survey and Beyond,” *Remote Sensing*, 17(2), p. 179. Available at: https://doi.org/10.3390/rs17020179.

IBM NASA Geospatial (2023) “Prithvi-100M-burn-scar.” Available at: https://doi.org/10.57967/HF/0953.

IBM NASA Geospatial (2023) “Prithvi-100M-sen1floods11.” Available at: https://doi.org/10.57967/HF/0973.

International Working Group on Satellite-based Emergency Mapping (IWG-SEM) (2015) *Emergency Mapping Guidelines v1.0*. Working Paper. Available at: https://www.un-spider.org/sites/default/files/IWG_SEM_EmergencyMappingGuidelines_v1_Final.pdf.

Jakubik, J. *et al.* (2025) “TerraMind: Large-Scale Generative Multimodality for Earth Observation.” arXiv. Available at: https://doi.org/10.48550/ARXIV.2504.11171.

Le Cozannet, G. *et al.* (2020) “Space-Based Earth Observations for Disaster Risk Management,” *Surveys in Geophysics*, 41(6), pp. 1209–1235. Available at: https://doi.org/10.1007/s10712-020-09586-5.

Lüddecke, T. and Ecker, A.S. (2021) “Image Segmentation Using Text and Image Prompts.” arXiv. Available at: https://doi.org/10.48550/ARXIV.2112.10003.

Ma, L. *et al.* (2019) “Deep learning in remote sensing applications: A meta-analysis and review,” *ISPRS Journal of Photogrammetry and Remote Sensing*, 152, pp. 166–177. Available at: https://doi.org/10.1016/j.isprsjprs.2019.04.015.

MMSegmentation Contributors (2020) “OpenMMLab Semantic Segmentation Toolbox and Benchmark.” Available at: https://github.com/open-mmlab/mmsegmentation.

Paszke, A. *et al.* (2019) “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” arXiv. Available at: https://doi.org/10.48550/ARXIV.1912.01703.

Portalés-Julià, E. *et al.* (2023) “Global flood extent segmentation in optical satellite images,” *Scientific Reports*, 13(1), p. 20316. Available at: https://doi.org/10.1038/s41598-023-47595-7.

Poursanidis, D. and Chrysoulakis, N. (2017) “Remote Sensing, natural hazards and the contribution of ESA Sentinels missions,” *Remote Sensing Applications: Society and Environment*, 6, pp. 25–38. Available at: https://doi.org/10.1016/j.rsase.2017.02.001.

Szwarcman, D. *et al.* (2024) “Prithvi-EO-2.0: A Versatile Multi-Temporal Foundation Model for Earth Observation Applications.” arXiv. Available at: https://doi.org/10.48550/ARXIV.2412.02732.

Wang, L. *et al.* (2022) “A review of the flood management: from flood control to flood resilience,” *Heliyon*, 8(11), p. e11763. Available at: https://doi.org/10.1016/j.heliyon.2022.e11763.

Wright, N. *et al.* (2025) “Training sensor-agnostic deep learning models for remote sensing: Achieving state-of-the-art cloud and cloud shadow identification with OmniCloudMask,” *Remote Sensing of Environment*, 322, p. 114694. Available at: https://doi.org/10.1016/j.rse.2025.114694.