The Journal of Open Source Software

IAMAP: Unlocking Deep Learning in QGIS for non-coders and limited computing resources

Paul Tresson [1¶], **Pierre Le Coz** [1,2], **Hadrien Tulet** [1], **Anthony Malkassian** [3], **and Maxime Réjou-Méchain** [1,2]

1 AMAP, Univ. Montpellier, IRD, CNRS, CIRAD, INRAE, Montpellier, France **2** Forest Restoration Research Unit, Department of Biology, Faculty of Science, Chiang Mai University, Chiang Mai, Thailand **3** Université de la Réunion, UMR PVBMT, St. Pierre, La Réunion, France ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 01 December 2025
Published: unpublished

Summary

1. Remote sensing has entered a new era with the rapid development of artificial intelligence approaches. However, the implementation of deep learning has largely remained restricted to specialists and has been impractical because it often requires (i) large reference datasets for model training and validation; (ii) substantial computing resources; and (iii) strong coding skills.

2. Here, we introduce IAMAP, a user-friendly QGIS plugin that addresses these three challenges in an easy yet flexible way. IAMAP builds on recent advancements in self-supervised learning strategies, which now provide robust feature extractors, often referred to as foundation models. These generalist models can often be reliably used in few-shot or zero-shot scenarios (*i.e.*, with little to no fine-tuning).

3. IAMAP's interface allows users to streamline several key steps in remote sensing image analysis: (i) extracting image features using a wide range of deep learning architectures; (ii) reducing dimensionality with built-in algorithms; (iii) performing clustering on features or their reduced representations; (iv) generating feature similarity maps; and (v) calibrating and validating supervised machine learning models for prediction.

4. By enabling non-AI specialists to leverage the high-quality features provided by recent deep learning approaches without requiring GPU capacity or extensive reference datasets, IAMAP contributes to the democratization of computationally efficient and energy-conscious deep learning methods.

Keywords: Remote sensing, Self-supervised learning, Foundation models, Machine learning, Artificial Intelligence, Consumer hardware.

Statement of need

The integration of remote sensing data with deep learning approaches is currently revolutionizing Earth observation sciences, leading to significant qualitative and quantitative improvements in large-scale predictions (Yasir et al., 2023; Yuan et al., 2020; Zhu et al., 2017). However, this revolution comes with a number of challenges. First, over the past decade, most deep learning applications have been highly data-demanding, requiring extensive manual labeling with typically more than one hundred thousands labeled points (Safonova et al., 2023). In most ecological and environmental science studies, constructing such a large reference dataset, through *e.g.*, ground observations or photo-interpretation, remains a major barrier to the implementation of deep learning approaches. Second, a common obstacle to the adoption of deep learning is the computing power required to train a model. Training a deep learning model is indeed highly resource-intensive, primarily due to the backpropagation step (see Goodfellow, 2016). As a result, modern deep learning architectures are virtually impossible to train without substantial local Graphics Processing Unit (GPU) capacity or access to high-end computing clusters. Last but not least, implementing deep learning approaches typically requires at least basic coding skills, which has so far restricted their use to users with a minimal background in computer science.

The recent development of self-supervised learning (SSL) approaches is a game-changer in the deep learning domain, as exemplified by the success of models like BERT and ChatGPT in natural language processing (Achiam et al., 2023; Devlin, 2018). In SSL, the model starts by learning features describing a dataset via a pretext task that does not require a label. In computer vision, several SSL strategies have been proposed, typically belonging to two main categories: contrastive or generative learning. In contrastive learning, several networks view transformed versions of the same data and have to learn to produce robust representation of this data (*e.g.* DINOv2 (Oquab et al., 2023) or VicReg (Bardes et al., 2021)). In generative learning, a network sees a degraded version of the data (typically, a masked version) and has to learn to generate a non-degraded version (*e.g.*, MAE, (He et al., 2022)) (for an overview of main SSL approaches, see Shwartz Ziv & LeCun, 2024). Once pre-trained on a large set of images, which remains very data- and resource-intensive, the resulting backbone can be referred to as a "foundation model". Like any pre-trained model, this foundation model can then be fine-tuned with a limited number of manually labeled examples to learn a specific downstream task (*e.g.* land cover classification or change detection in remote sensing) (Ericsson et al., 2021). The main difference between a pre-trained self-supervised learning (SSL) model and a pre-trained supervised model lies in their training objectives: SSL models are not constrained by predefined labels and are therefore free to explore and encode the intrinsic structure and diversity of the data, often resulting in more general and transferable representations. In contrast, supervised models are explicitly optimized to perform a specific user-defined task, which can lead to highly specialized representations that may overlook other meaningful features in the data. As such, SSL foundation models can perform well even in low-shot or zero-shot tasks, *i.e.* using the model as is, with few or no training data. Consequently, SSL models are considered particularly promising for remote sensing tasks, as demonstrated by recent works and initiatives (Cong et al., 2023; Jakubik et al., 2023; Marsocci et al., 2024; Xiong et al., 2024).

In parallel, to the development of SSL, Vision Transformers (ViT) (Dosovitskiy et al., 2020) and their derivatives (such as EVA (Fang et al., 2023) or Hiera (Ryali et al., 2023)) have changed the state of the art of computer vision. In a ViT, an image is analyzed by patches (usually 16×16 or 14×14 pixels). Each patch is projected in an embedding space and the embedding spaces of the different patches update each-others through the self-attention mechanism (see Vaswani, 2017). These architectures present the advantage that the features produced at patch level can be analyzed spatially within an image, which is relevant in remote sensing especially when working at high resolution, (see (Marsocci et al., 2024)) (see the plugin documentation for a more detailed overview of the functioning of a ViT).

With the democratization of deep learning, some developers have already worked on the integration of deep learning models in geographic information systems such as the open-source and widely used QGIS software (QGIS Development Team, 2025). However, at the time of writing, these solutions mostly focus on fine-tuning models or using a model in inference only (*e.g.* see Aszkowski et al., 2023; Zhao et al., 2023). Then, they are only usable by users with access to high-end computing power, extensive dataset, on interested in a task for which a specific model was already trained.

In this paper, we introduce a new plugin for QGIS designed to streamline remote sensing image analysis using advanced pre-trained deep learning models without the need for coding or extensive computing resources. As demonstrated in this paper, users can apply a pre-trained model to generate high-quality features at the patch level. The plugin then allows users to manipulate these features using various projections, clustering, similarity, and supervised machine learning (ML) algorithms.

Plugin description

The IAMAP plugin integrated into QGIS consists of five main modules, which can be used individually or sequentially on a georeferenced raster image (Fig. Figure 1. We here below describe the functionality of each module.

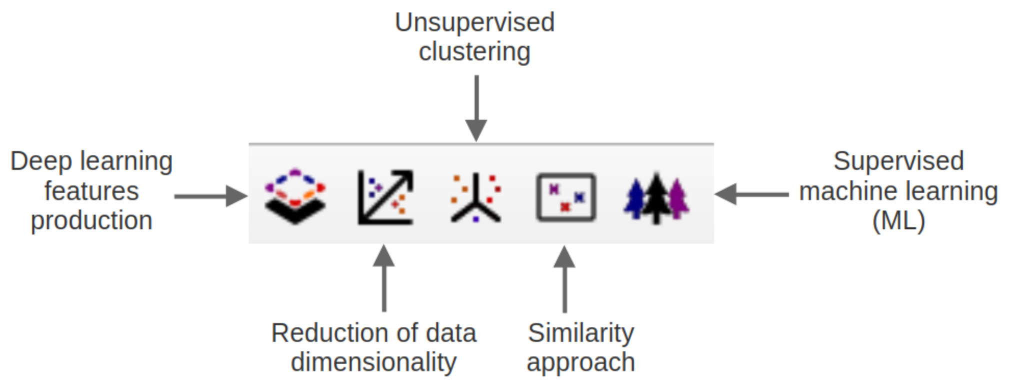
Unsupervised clustering
Deep learning features production
Supervised machine learning (ML)
Reduction of data dimensionality
Similarity approach

Figure 1: The five main modules of the IAMAP plugin.

Deep Learning feature production

The first and most original module of IAMAP is the deep learning feature extraction module. Using a georeferenced raster as input (a QGIS raster layer or a raster saved on disk), this module enables the use of various pre-trained deep learning models to produce a set of features describing the input raster. The use of deep learning model in inference only removes the costly training step and greatly reduce the computational power required. This module mostly relies on two widely used *pytorch* libraries: *timm* (Wightman, 2019), for loading pre-trained model weights, and *torchgeo* (Stewart et al., 2022), for handling geospatial data.

The *timm* library has become a standard for sharing and loading pre-trained weights in *pytorch* and is now integrated into the HuggingFace Hub(Wolf, 2019). Originally developed for sharing natural language processing (NLP) models, the HuggingFace Hub has since become the largest repository of pre-trained deep learning models, with over 400,000 models available at the time of writing. Our aim in choosing this back-end is to rely on libraries that are widely used, well maintained, and actively updated. Hence, while we propose a couple of widely used foundation models by default, the user can select any model available on HuggingFace by entering the architecture name (although not all models are guaranteed to work depending on their architecture). The plugin interface also gives the possibility to load local pre-trained models weights, if a correct *timm* architecture is chosen.

Handling remote sensing datasets differs from working with typical image collections used in classical computer vision. Raster images are often several orders of magnitude larger and must be sampled to fit the input requirements of neural networks, which typically expect square images a few hundred pixels wide. Additionally, it is essential to preserve the geographical metadata associated with each raster. The *torchgeo* library provides an efficient solution to address these constraints, but it includes many features and dependencies that are not all useful for our purposes. Therefore, we have forked only the necessary parts of the *torchgeo* code into our plugin. Our goal is to keep the codebase simple and minimize unnecessary dependencies.

The module offers several options to the user, most of which come with proposed default values. Among these, the sampling size and the stride are key parameters: the sampling size determines the dimensions of the extracted tiles while the stride controls the spacing between tiles and thus the degree of overlap used to reduce tiling artifacts. The combination of sampling size, stride and the architecture chosen as encoder will determine the resolution of the output raster. It is possible as well to set an overlap between tiles to reduce possible tilling effects. These parameters are essential to consider, as they directly influence the trade-off between model performance and inference cost.

The output of this module is a raster with a coarser resolution than the input raster, depending on the sampling parameters and the chosen deep learning architecture. It contains as many bands as the number of extracted features (*e.g.* 768 for a ViT-base model). By default, QGIS loads the raster at the end of the process and displays only the first three bands using a false-color RGB composition, although these bands are not necessarily the most informative (see top row of Fig. Figure 2).

Computer vision state of the art pretrained models are usually trained with Red Green and Blue (RGB) bands used in natural images. We thus propose three strategies for users who want to work with input rasters with a band number different from 3, as it is usually the case in remote sensing. One potential solution requires manipulations of the pre-trained weights to handle the exact number of bands of the raw image by either copying the weights of the first layer modulo 3 if the number of input bands is larger than three or averaging weights if the number of input bands is smaller than three. This option should be taken with caution given that it is expected to change the behavior of the model, even if it should keep a capacity for abstraction and projecting low level information into a richer feature space (see Fig. Figure 2 examples). The second solution consists in selecting only 3 relevant bands in the deep learning module without modifying the model's weights. The last option, which appears to be the most robust one according to our tests, consists in applying first a dimension reduction (see next section), such as a PCA, and use three reduction axes as an input in the deep learning encoder.

As the state of the art is evolving for remote sensing application, we also provide inference with foundation models trained specifically on remote sensing data, such as DOFA (Xiong et al., 2024) and a ViT pretrained on the SSL4EO dataset (Wang et al., 2022) using Marsocci et al. (2024) implementation (Marsocci et al., 2024). For the DOFA encoder, multispectral bands are handled by the model without manipulation of the encoder or pre-processing required.

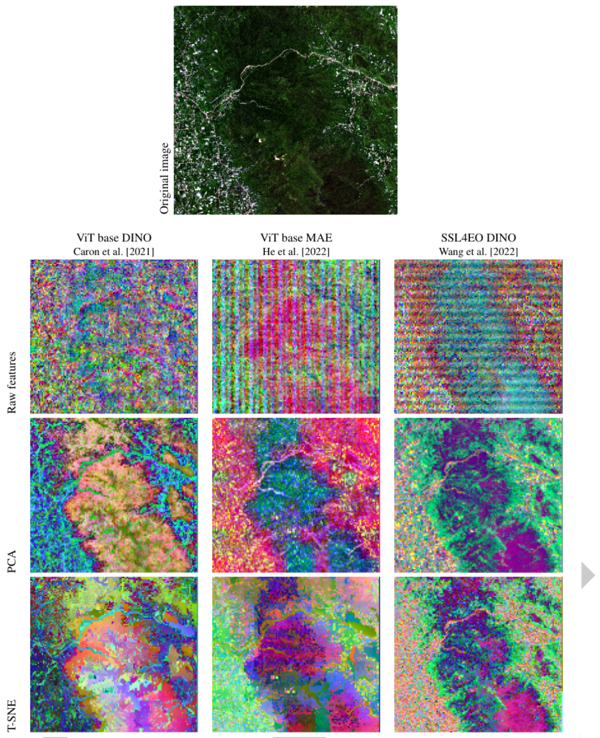

Figure 2: A sentinel 2 image of a forested landscape in Thailand (Khao Banthat Wildlife Sanctuary; Lat 7.53°, Lon 99.82°) processed by different backbones. The top row represents the first three feature dimensions output by the models (which may not be the most informative). The second row shows a 3D PCA of the features mapped to the red, green and blue channel respectively. The third row shows a projection using a 3D T-SNE.

Reduction of data dimensionality

This module enables dimensionality reduction of an input raster using a variety of approaches, including PCA, t-SNE (Van der Maaten & Hinton, 2008), and UMAP (McInnes et al., 2018). This dimensionality reduction step is particularly useful for two tasks: (1) reducing the number of bands in a raw multi-band raster before applying a deep learning model, as discussed in the previous section, and (2) reducing the dimensionality of the feature space to facilitate visualization and support more robust training procedures. Indeed, deep learning models typically produce a high-dimensional feature space. While this high dimensionality poses no issues when fed into a deep learning head, it can become a drawback for visualizing the feature space and using it in lighter machine learning models such as Random Forests. To address this, it is common in deep learning research to use dimensionality reduction algorithms to visualize and analyze the feature space of a model. These reduced features can often be more informative at first glance (see the second row of Fig. Figure 2), and reducing or ordering the input dimensions can improve the performance of other algorithms afterward (see the third row of Fig. Figure 2).

This module relies on the *scikit-learn* library, which provides access to a wide range of algorithms (25 at the time of writing). As a result, all algorithms available in the *scikit-learn* `decomposition` and `cluster` modules that have common APIs (namely, a `fit()`, a `transform()`, or a `fit_transform()` method) can be used. Note that the UMAP approach relies instead on its dedicated Python implementation and is an optional dependency at the time of writing.

Unsupervised clustering

A common operation when handling feature spaces is clustering to assign classes to data points. The unsupervised clustering module allows to implement various unsupervised clustering algorithms, including K-means or HDBSCAN (McInnes et al., 2017) (see Fig. Figure 3). This module again relies on *scikit-learn* as a back-end. As such, all algorithms available in the *scikit-learn* `cluster` module sharing common APIs (namely, a `fit()`, a `predict()`, or a `fit_predict()` method) can be used.

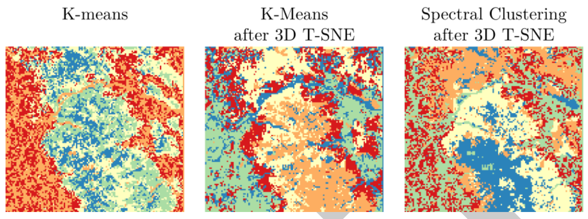

Figure 3: Example of different clustering (k=5) of the ViT Base DINO features.

Similarity approach

When exploring high-dimensional spaces, similarity search is a common task. The similarity approach module of IAMAP enables users to generate similarity maps based on one or more point shapefiles. This module relies on cosine similarity, which assigns a score between 0 and 1 to two points based on their coordinates in the feature space. The score is zero if the vectors represented by these coordinates are orthogonal to the reference vectors provided by the user, and 1 if they are identical. This approach is commonly used for instance retrieval tasks in deep learning (Chen et al., 2022), as it helps identify points that are closely represented in the feature space (see Fig. Figure 4 for examples). By applying a threshold, this method can also be used for simple segmentation tasks.

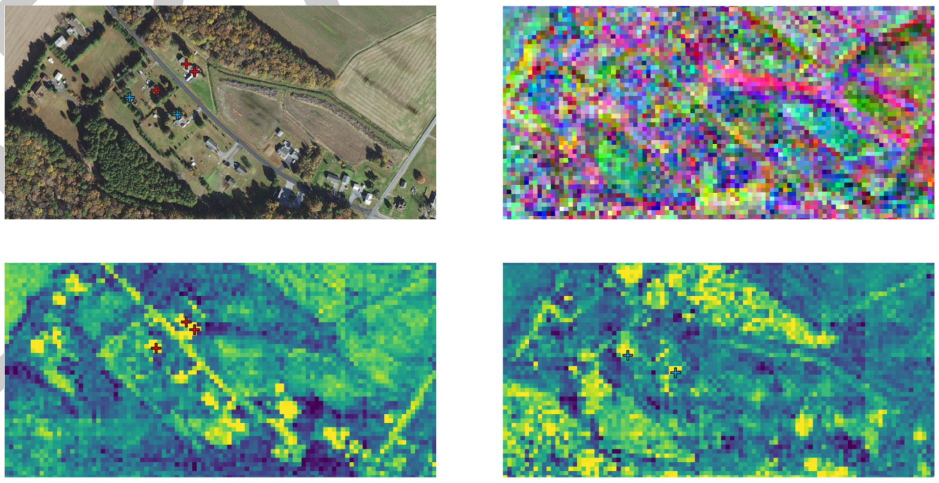

Figure 4: Example usage of cosine similarity with NAIP data. Using only a couple of points, we can identify trees or houses without needing to train a model for this specific task. From left to right and top to bottom: Original RGB data and provided template points (red and blue crosses); Features produced by a ViT DINO small encoder (Caron et al., 2021); Heatmap produced with the red points as input (houses); Heatmap produced with the blue points as input (trees with red leaves).

Supervised machine learning

The final module of IAMAP enables users to build supervised predictive models using classical machine learning algorithms such as Random Forests, KNN, or Gradient Boosting. In contrast to other plugins that enable the use of end-to-end deep learning models for specific tasks (*e.g.* Aszkowski et al., 2023), we have focused on lighter machine learning algorithms to minimize dataset and computational resource requirements. These algorithms often require relevant input features to be able to perform. While deep learning is now more potent on a lot of tasks, ML algorithms used with deep learning features as input can achieve satisfactory performances with a fraction of the cost needed to fit the algorithm. Then for example, algorithms such as KNN are used in deep learning research to evaluate models trained in an unsupervised way without having to retrain an entire model (*e.g.* see SM of (Caron et al., 2021)).

The plugin provides a wide array of available algorithms, once again using the *scikit-learn* library as a back-end. More specifically, all methods provided by the `ensemble` and `neighbors` modules that share a common API are available.

Because this module relies on supervised approaches, it requires the user to provide a reference point dataset as a shapefile. The sampling design of this reference dataset is left to the user but we provide the option to choose how the validation scheme is performed. By default, a cross validation in k-fold is performed by randomly splitting the dataset into 5 folds. Otherwise, it is possible to define the train/test split or the cross-validation scheme dataset according to the values in a attribute column. As the appropriate validation scheme depends largely on the dataset and target task, this validation scheme might not be the most appropriate (see (Ploton et al., 2020) for discussion on this topic in the context of spatial datasets). We therefore encourage the users to consider their choices of validation scheme via the plugin interface.

Usage example

Because the IAMAP plugin consists of a set of different modules that can be implemented independently or sequentially in various combinations, the number of possible uses is very large. Here, we provide one example of a potential workflow to produce a classification map using three complementary IAMAP modules (Fig.Figure 5). Several other use cases, along with detailed protocols, are available in the online documentation of the plugin.

An example of workflow implemented using IAMAP to produce a ca. 50-m classification map from a 10-m multispectral Sentinel 2 image over a forested landscape from Thailand (Lat 7.53°, Lon 99.82°).

Figure 5: An example of workflow implemented using IAMAP to produce a ca. 50-m classification map from a 10-m multispectral Sentinel 2 image over a forested landscape from Thailand (Lat 7.53°, Lon 99.82°).

Design choices

We have aimed the development for the plugin to be usable on a laptop without a GPU by someone with no coding experience. This has come with various design choices.

Cross-platform and easy to install

The plugin is designed to be easy to install, especially working with state of the art deep learning dependencies. Dependencies are handled using *pip* with a startup script that automatically looks for dependencies and installs the missing one if needed. If needed, a *conda* environment formula is provided as well to work in a separate fixed environment. The plugin has been tested on Windows, Mac and Linux with several QGIS versions. Although the plugin does not require a GPU to function, if one is detected, the correct version of *pytorch* is downloaded to be used during deep learning inference. The user may opt out the usage of the GPU afterwards.

Inference as a stoppable background task

The inference of deep learning models on large raster images may be long, in particular without a GPU. Then, we have given the option to schedule small pauses during the inference, which limits the CPU usage and enables to use the PC for other tasks during the inference. An other choice has been to save batches on disk rather than keeping all inferred tiles on RAM. While slower, this makes possible to stop the inference and start again latter (even after reboot). Temporary files are cleaned up after use.

The produced rasters can become heavy and are therefore compressed by default to save space.

Model quantization

The quantization of a deep learning model is the act of switching the encoding of the weights from *float32* to a lighter format such as *uint8*. This greatly reduces model size and inference time, at the cost of some precision (see pytorch documentation) (Wu et al., 2020). Recently, DeepSeek AI have been able to divide training costs by 40 by relying on similar methods with *fp8* precision training (Liu et al., 2024). This practice is common when working with hardware size constraints. Here, we give the option to the user to quantize the model before inference. When working with a model that was not specifically trained for the task asked of it, the trade-off between speed and precision could be beneficial more often than not.

Perspectives and future developments

Limitations of the plugin

This plugin is though for a usage in conditions where the end-to-end training of a neural network is not a possibility because of a lack of data or computing power. This comes with limitations to what is possible with deep learning in inference only compared to what can be achieved with neural network trained classically.

First, some task will require non-linear and complex connections in the feature space and will not be possible with simple manipulations as those possible with this plugin. For example, complex tasks as instance segmentation is easily achievable with dedicated deep learning models (see (Zhao et al., 2023)) but not with our plugin.

By using deep learning methods, classical machine learning and data manipulation methods, this plugin inherits from advantages but also drawbacks from different types of algorithms. Ideally and depending on the use case, the use of a deep learning encoder will provide relevant features, robust to low level noise and transformations. These features can then be leveraged with lighter machine learning algorithms, enabling the creation of maps that would not be possible without the features provided by a deep learning encoder. On the other hand, it may be required to test a variety of encoders and hyper-parameters to achieve satisfying results. While projection or clustering techniques are often easy to fit, testing different deep learning models can be time consuming, especially on restricted hardware.

Future developments

Future developments for the plugin include keeping up with computer vision state of the art but also optimization techniques to ensure lightweight inference time and usability on restricted hardware.

Moreover, we aim to implement more models dedicated to remote sensing tasks (for instance, those evaluated by (Marsocci et al., 2024)). As of now, the feature extraction tool is though for ViT like encoders, that have spatially explicit features. We aim to develop it to be more generalist and take any encoder as input, such as ResNets or UNets that are still widely used in deep learning and remote sensing.

Availability

Development of the plugin is open sourced on GitHub. Documentation is available on readthedocs. The plugin is developed in continuous integration. We plan to publish the plugin on official QGIS repository to further ease the installation process.

Acknowledgments

The authors would like to thank all people who have tested this software during development and have provided meaningful feedback.

Conflict of interest

The authors declare no conflict of interest.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., & others. (2023). Gpt-4 technical report. *arXiv Preprint arXiv:2303.08774*.

Aszkowski, P., Ptak, B., Kraft, M., Pieczyński, D., & Drapikowski, P. (2023). Deepness: Deep neural remote sensing plugin for QGIS. *SoftwareX*, *23*, 101495. https://doi.org/https://doi.org/10.1016/j.softx.2023.101495

Bardes, A., Ponce, J., & LeCun, Y. (2021). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv Preprint arXiv:2105.04906*.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). Emerging properties in self-supervised vision transformers. *Proceedings of the International Conference on Computer Vision (ICCV)*, 9650–9660.

Chen, W., Liu, Y., Wang, W., Bakker, E. M., Georgiou, T., Fieguth, P., Liu, L., & Lew, M. S. (2022). Deep learning for instance retrieval: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(6), 7270–7292.

Cong, Y., Khanna, S., Meng, C., Liu, P., Rozi, E., He, Y., Burke, M., Lobell, D. B., & Ermon, S. (2023). *SatMAE: Pre-training transformers for temporal and multi-spectral satellite imagery*. https://arxiv.org/abs/2207.08051

Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv Preprint arXiv:1810.04805*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., & others. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv Preprint arXiv:2010.11929*.

Ericsson, L., Gouk, H., & Hospedales, T. M. (2021). How well do self-supervised models transfer? *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5414–5423.

Fang, Y., Wang, W., Xie, B., Sun, Q., Wu, L., Wang, X., Huang, T., Wang, X., & Cao, Y. (2023). Eva: Exploring the limits of masked visual representation learning at scale.

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 19358–19369.

Goodfellow, I. (2016). *Deep learning*. MIT press.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022). Masked autoencoders are scalable vision learners. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16000–16009.

Jakubik, J., Roy, S., Phillips, C. E., Fraccaro, P., Godwin, D., Zadrozny, B., Szwarcman, D., Gomes, C., Nyirjesy, G., Edwards, B., Kimura, D., Simumba, N., Chu, L., Mukkavilli, S. K., Lambhate, D., Das, K., Bangalore, R., Oliveira, D., Muszynski, M., … Ramachandran, R. (2023). Foundation Models for Generalist Geospatial Artificial Intelligence. *Preprint Available on Arxiv:2310.18660*.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., & others. (2024). Deepseek-v3 technical report. *arXiv Preprint arXiv:2412.19437*.

Marsocci, V., Jia, Y., Bellier, G. L., Kerekes, D., Zeng, L., Hafner, S., Gerard, S., Brune, E., Yadav, R., Shibli, A., Fang, H., Ban, Y., Vergauwen, M., Audebert, N., & Nascetti, A. (2024). *PANGAEA: A global and inclusive benchmark for geospatial foundation models*. https://arxiv.org/abs/2412.04204

McInnes, L., Healy, J., Astels, S., & others. (2017). Hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, *2*(11), 205.

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv Preprint arXiv:1802.03426*.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., & others. (2023). Dinov2: Learning robust visual features without supervision. *arXiv Preprint arXiv:2304.07193*.

Ploton, P., Mortier, F., Réjou-Méchain, M., Barbier, N., Picard, N., Rossi, V., Dormann, C., Cornu, G., Viennois, G., Bayol, N., & others. (2020). Spatial validation reveals poor predictive performance of large-scale ecological mapping models. *Nature Communications*, *11*(1), 4540.

QGIS Development Team. (2025). *QGIS geographic information system*. Open Source Geospatial Foundation. http://qgis.org

Ryali, C., Hu, Y.-T., Bolya, D., Wei, C., Fan, H., Huang, P.-Y., Aggarwal, V., Chowdhury, A., Poursaeed, O., Hoffman, J., Malik, J., Li, Y., & Feichtenhofer, C. (2023). Hiera: A hierarchical vision transformer without the bells-and-whistles. *ICML*.

Safonova, A., Ghazaryan, G., Stiller, S., Main-Knorn, M., Nendel, C., & Ryo, M. (2023). Ten deep learning techniques to address small data problems with remote sensing. *International Journal of Applied Earth Observation and Geoinformation*, *125*, 103569.

Shwartz Ziv, R., & LeCun, Y. (2024). To compress or not to compress—self-supervised learning and information theory: A review. *Entropy*, *26*(3), 252.

Stewart, A. J., Robinson, C., Corley, I. A., Ortiz, A., Lavista Ferres, J. M., & Banerjee, A. (2022). TorchGeo: Deep learning with geospatial data. *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, 1–12. https://doi.org/10.1145/3557915.3560953

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11).

Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

Wang, Y., Braham, N. A. A., Xiong, Z., Liu, C., Albrecht, C. M., & Zhu, X. X. (2022). SSL4EO-S12: A large-scale multi-modal, multi-temporal dataset for self-supervised learning in earth observation. *arXiv Preprint arXiv:2211.07044*.

Wightman, R. (2019). PyTorch image models. In *GitHub repository*. https://github.com/rwightman/pytorch-image-models; GitHub. https://doi.org/10.5281/zenodo.4414861

Wolf, T. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv Preprint arXiv:1910.03771*.

Wu, H., Judd, P., Zhang, X., Isaev, M., & Micikevicius, P. (2020). Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv Preprint arXiv:2004.09602*.

Xiong, Z., Wang, Y., Zhang, F., Stewart, A. J., Hanna, J., Borth, D., Papoutsis, I., Saux, B. L., Camps-Valls, G., & Zhu, X. X. (2024). Neural plasticity-inspired foundation model for observing the Earth crossing modalities. *arXiv Preprint arXiv:2403.15356*.

Yasir, M., Jianhua, W., Shanwei, L., Sheng, H., Mingming, X., & Hossain, M. (2023). Coupling of deep learning and remote sensing: A comprehensive systematic literature review. *International Journal of Remote Sensing*, *44*(1), 157–193.

Yuan, Q., Shen, H., Li, T., Li, Z., Li, S., Jiang, Y., Xu, H., Tan, W., Yang, Q., Wang, J., & others. (2020). Deep learning in environmental remote sensing: Achievements and challenges. *Remote Sensing of Environment*, *241*, 111716.

Zhao, Z., Fan, C., & Liu, L. (2023). *Geo SAM: A QGIS plugin using Segment Anything Model (SAM) to accelerate geospatial image segmentation* (Version 1.1.0). Zenodo. https://doi.org/10.5281/zenodo.8191039

Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine*, *5*(4), 8–36.