DeepSatData: Building large scale datasets of satellite images for training machine learning models

Michail Tarasiou and Stefanos Zafeiriou

***Abstract*—This paper presents *DeepSatData* a pipeline for automatically generating satellite imagery datasets for training machine learning models. We also discuss design considerations with emphasis on dense classification tasks, e.g. semantic segmentation. The implementation presented makes use of freely available *Sentinel-2* data which allows the generation of large scale datasets required for training deep neural networks (DNN). We discuss issues faced from the point of view of DNN training and evaluation such as checking the quality of ground truth data and comment on the scalability of the approach. Accompanying code is made publicly available in https://github.com/michaeltrs/DeepSatData.**

I. Introduction

Currently there are more than 150 satellites in orbit equipped with dedicated instruments gathering data for a variety of Earth Observation (EO) tasks. An ever increasing amount of that data are made freely accessible to the public, for example approximately $20Tb$ of new data are made available every day just through the European Space Agency's Sentinel 1-3 satellites.

The *Copernicus Open Access Hub* (COAH) provides free and open access to data captured by the *European Space Agency's* Sentinel missions starting from the In-Orbit Commissioning Review (IOCR). These data are made available directly through COAH either by use of a graphical user interface [1], through a variety of platforms from the Copernicus Data and Information Access Services (DIAS) [2], [3], [4], [5], [6] or through mirror sites [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17].

However, there are not, to the best of our knowledge, publicly available tools automating the entire process from downloading to processing Sentinel products at a scale required for successfully training machine learning models. In this paper we present *DeepSatData* a simple tool for downloading and processing Sentinel products from the point of view of DNN training. With *DeepSatData* it is possible to automatically download available satellite imagery for a given area of interest (AOI) and time period of interest (POI) and to couple these with available ground truth data to create fully annotated datasets. In addition we present some general considerations for generating satellite imagery datasets suitable for DNN training. Particular emphasis is placed on dense classification for agricultural data, e.g. crop type semantic segmentation, however, the structure of the provided code is general enough to accommodate different types of geodata and remote sensing tasks.

II. Background

A. Densely annotated data

Typically, the anatomy of a dense classification dataset for computer vision involves input arrays and dense annotations matching two or more dimensions of the inputs. In general obtaining annotations for dense classification tasks is a time consuming process, for example it is estimated that annotating a single image from the *Cityscapes* dataset [18] fine set takes approximately 90min of work. For datasets in which annotations are not included for all objects found in the inputs it is common practice to assign all multiple unknown objects into a single class which is either treated as an unknown or as part of a *background* class. Depending on the formulation of the task it is possible to treat the *background* class as another regular class or mask its influence during training and only learn to recognise the remaining classes.

B. Dense classification tasks

Similar to the general classification problem where the goal is to assign one of N known classes to an input array, dense classification aspires to assign a class to every location, e.g. pixel, of an input array. Distinguishing between the different types of input arrays and the type of information encoded by the output classes can lead to defining several problems in computer vision. Inputs in general contain 2 or 3 spatial dimensions or a time dimension each with a fixed number of channels. For satellite imagery we are interested in either 2D images, i.e. a single image, or time series of images. In the second case each image is typically accompanied by a time stamp indicating the image capture time. The interval between successive satellite image captures at the same location is generally not constant. This is in contrast to video data, also consisting of time series of 2D images, in which there is a fixed time-step between successive frames, the inverse of the frame rate. The model output most commonly encodes semantic or identity information or both leading to the tasks of *semantic segmentation* [19], [20], [21], [22], [23], [24], [25], [26], [27], *instance segmentation* [28], [29], [30], [31], [32], [33], [34] and *joint semantic-instance segmentation* [35], [36], [37].

III. Downloading satellite data

Downloading all required data for an AOI and POI can be a lengthy process. That is particularly the case for data captured more than 12 months in the past which will need to be accessed through the COAH's Long-Term Archive (LTA). This means that the data will first have to be requested by the LTA and will be made available to download within after some time has elapsed (24h). Additionally, there is a maximum allowed number of requests per user to the LTA at a rate of one product request every 30min. In fact when working with annotated data it is most likely that these correspond to a period in the past thus all imagery products will need to be downloaded following that process. The limit in the amount of data that can be requested by the LTA poses a hard constraint on the number of products that can realistically be downloaded forcing us to optimize our product selection process. Given the importance of selecting the right products in space and time we propose to perform the selection process manually and automate the remaining part of the dataset generation process. Below are some general criteria for optimizing the product selection process.

A. *Low cloud cover ratio*

Cloud cover percentage is calculated for the full spatial extent of a Sentinel product. While it can be the case that a clear view of the AOI can be found in a cloudy image, especially so for a small AOI, it is likely to get more clear images from products with low cloud coverage. Thus, we prioritize downloading the less cloudy products first.

B. *Large overlap with the AOI*

Each Sentinel-2 tile covers a region of around 100km $\times$ 100km which is large enough such that a single tile can be used for a dataset. For example using striding windows of 240m $\times$ 240m (24×24 pixels for the largest resolution band) results in approximately 200k samples. For a small AOI it is likely that it can be covered by a single Sentinel tile in which case there is 100% coverage of the AOI by that tile. Otherwise more than one products will need to be downloaded to cover the full extent of the AOI, in which case it is convenient to start with the products that cover the largest part of the AOI.

C. *Large product size*

As described in the S2 product description website "Tiles can be fully or partially covered by image data. Partially covered tiles correspond to those at the edge of the swath.". In products partially covered with image data only part of the image contains information with remaining part covered by zero values. We prioritize downloading products with a small proportion of zero valued regions and make use of the product size (Mb) as a proxy for the degree of coverage.

D. *Uniformly spread along the time period of interest*

Modern EO satellites can have a very small revisit time. For example the two satellites which form the Sentinel-2 constellation can have a revisit time of as few as 5 days. Rather than downloading products for all available dates during a POI we may need to sample from available dates. Unless otherwise required by experimental settings we choose to select products such that they are spread as uniformly as possible during the POI.

IV. Data generation pipeline

Having downloaded a set of satellite imagery products what is of interest is to extract small image patches of constant size that can fit into hardware accelerator memory. When working with temporal tasks it is also of interest to group/sort these patches by location into time series objects that can be used to train temporal models. Fig.1 shows a schematic overview of this process and also indicates the relative size of typically extracted patches compared to the size of downloaded satellite products. Depending on whether there are available ground truth annotations we may choose to only process locations for which there are ground truths. These steps are further elaborated in the following sections.

A. *From vector to raster ground truth data*

This step is only relevant for cases when there are available ground truth annotations. We assume these collections are in the form of geo-polygons whose vertices are GPS coordinates at a given coordinate reference system (CRS) as this is the way most commonly used to denote the location of objects on the Earth's surface. To ensure consistency we define a canonical form of representing such collections which includes the following fields for each object:

- `geometry` is a geo-polygon containing GPS coordinates for all the vertices of the object.
- `crs` denotes the geographic CRS used.
- `ground_truth` indicates the ground truth value corresponding to area defined by `geometry`. This is typically of type `int` for semantic or identity classes and type `float` in the case of regression tasks.
- `year` denotes the time period the ground truth is valid for the given geometry. This value need not necessarily correspond to a calendar year, any distinct value will be used to group observations corresponding to the same time period.

Using these data we follow a rasterization step. Here we first define a grid which is initiated by a value corresponding to the background. For classification tasks this will be the value of the *background class*. For each pixel in the grid we calculate the ratio of the pixel area that is covered by the geo-polygon. All pixels partly or fully covered by the geo-polygon are assigned the `ground_truth` corresponding to that polygon. Additionally, each polygon is assigned a unique id which is used to create a raster map for object identities using the same process as above. This could be used either for instance segmentation like tasks or as a means to aggregate information during evaluation, e.g. calculate performance metrics with respect to the size or the geometry of objects. It is typical to define the grid size equal to the largest resolution satellite image available, however, this need not necessarily be the case. For example using CNNs it is straightforward to control the output size of our model to match any rasterization resolution of choice. Also, for crop-type semantic segmentation [38] showed that it is possible to successfully learn to distinguish crop types at a higher resolution than satellite image pixels. An example of performed rasterization is shown in Fig.2.

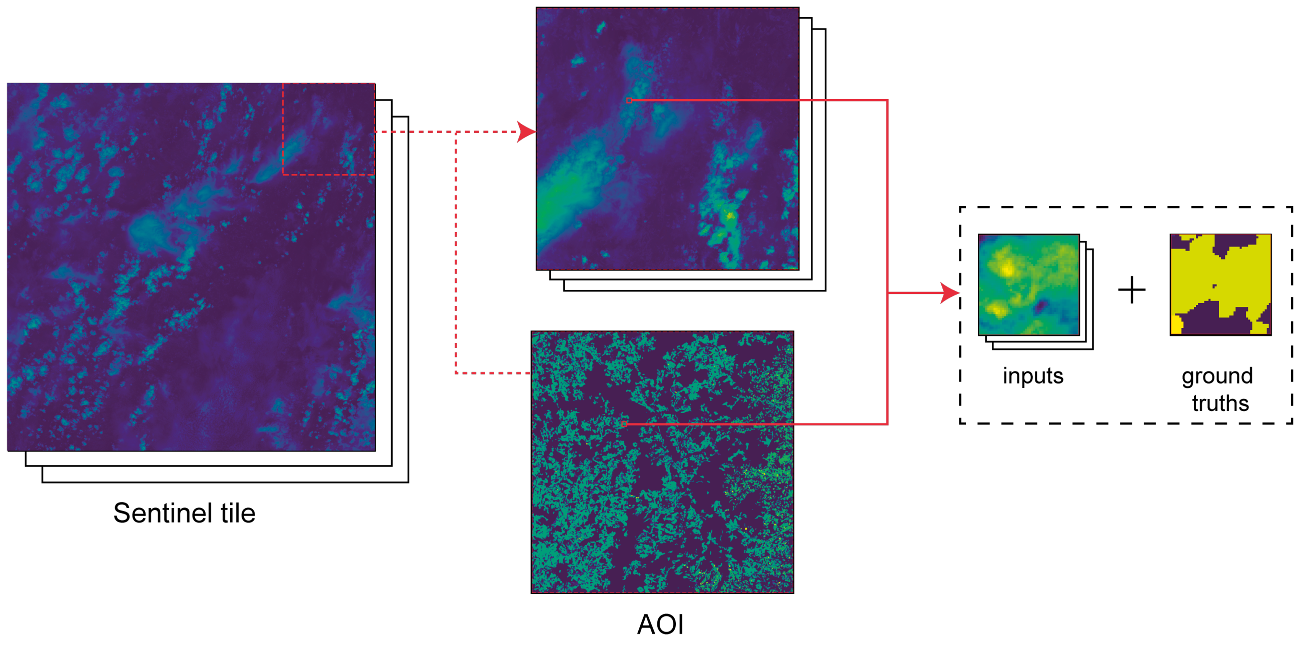

Figure 1. Data generation process with ground truth annotations. Satellite product shown here (left) is a Sentinel-2 tile which covers a 100×100 km^2 area at a resolution of 10 m. Extracted sample size (right) covers a 0.5×0.5 km^2 area.

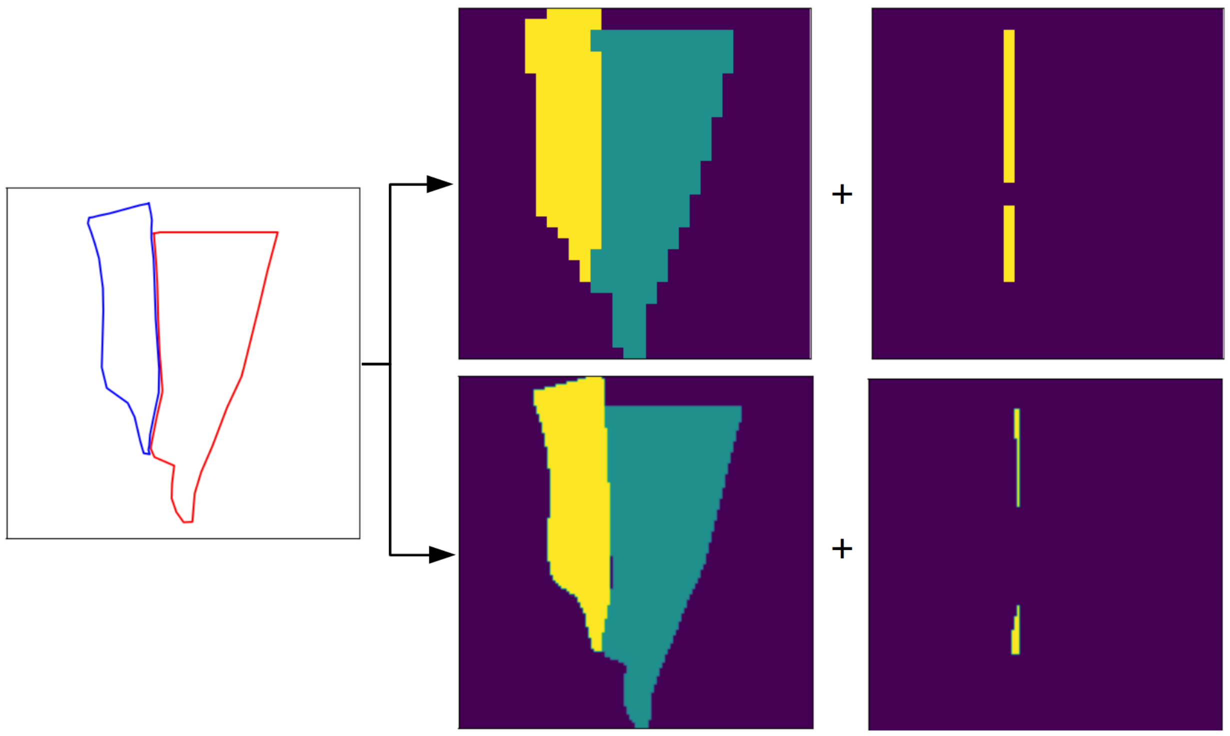

Figure 2. Parcel rasterization. Parcel vector geometries (left) are rasterized to get a ground truth class map (center) and doubly assigned pixel masks (right). Two different resolutions shown at 10m (top) and 2.5m (bottom). We note how the high resolution raster image contains a smaller ratio of doubly assigned pixels.

1) Masking ground truth inconsistencies: The process of generating dense ground truth annotations for geodata is unique w.r.t other dense labelled data, e.g natural images [18], [39], [40], in that source images and ground truths are first collected separately and are then aligned by location. While a human annotator working on a semantic segmentation dataset will draw semantic classes on top of captured images, ground truth collection for remote sensing involves a step of gathering GPS coordinates on the field and a separate step of matching these with source images. This introduces the possibility for

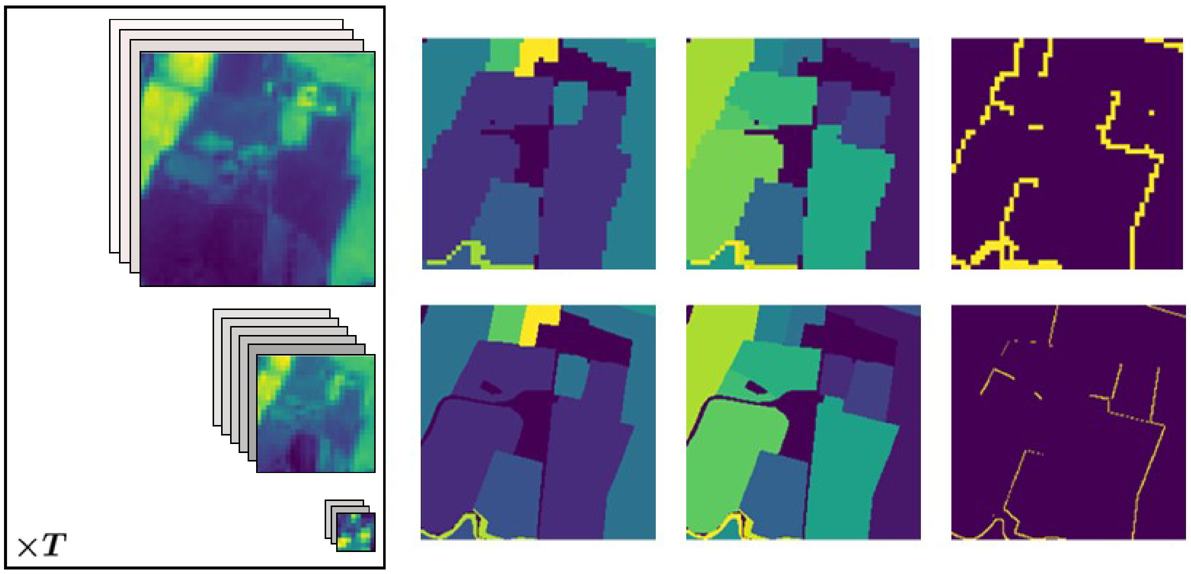

Figure 3. Example of data extracted using the proposed pipeline. (left) Satellite image inputs. Here shown all four 10m bands, six 20m bands and three 60m bands for a variable number of time steps T depending on location. (right) Ground truth data. From left to right we show crop types, parcel ids and doubly assigned pixels. Shown at 10m resolution (top row) and 2.5m resolution (bottom row). If there are no available ground truths only the input data are extracted (left).

systematic geolocation errors, the gathered GPS coordinates might not be in complete agreement with the geolocation corresponding to the satellite images. While it is possible to identify some cases where there are noticeable offsets between inputs and ground truths by inspection, [41] identify single pixel offset errors, in general it is impossible for a human to correct all such mistakes. For this reason we are using boolean masks to mark inconsistencies when it is possible to identify, such as pixels that fall inside multiple polygons during the rasterization step. We distinguish between pixels that are partly or fully claimed by two or more polygons. While the former case is a natural outcome of the rasterization step and is improved with using a higher resolution grid, as shown in Fig.2 low vs high resolution, the latter case clearly indicates a geocoding error in either polygon.

B. Splitting a Sentinel product to small windows and making time series objects

The main reason for choosing to split a satellite imagery product into smaller, equal size patches is the requirement to load multiple time series objects into hardware accelerator memory for efficient training. While the size of a satellite image is in the order of tens or hundreds of km, e.g a single Sentinel-2 tile covers a 100×100 km^2 area, sizes typically used for semantic segmentation are in the order of hundreds of m, e.g $240m$ [41], $480m$ [41], [38], $640m$ [42]. An example of that scale difference can be seen in Fig.1. We may choose to split the AOI only for locations where ground-truth annotations are available or, as is the case for unsupervised learning tasks, we may choose to split the entire AOI.

The end result of the data generation process is a set of time series objects containing image patches corresponding to the same location at different time stamps. Even though it would be possible to load all satellite images for all time stamps in memory and split-save to disk in one step this can be forbidding in terms of memory consumption for large AOI and POI. For this reason we choose to separate the steps of extracting patches and grouping/sorting these in time to create the final outputs. An example of data included in a single sample point extracted using the *DeepSatData* pipeline can be seen in Fig.3.

V. Conclusion

This paper presented *DeepSatData* a pipeline for automatically generating data for training machine learning models on earth observation tasks and explained the main considerations behind its design. While particular emphasis was placed on generating datasets for dense classification tasks using time series of satellite images it is trivial to extend the provided code to extract single images for dense or global classification. We intend to provide such capabilities in future updates.

References

[1] "SNAP-ESA sentinel application platform v2.0.2," http://step.esa.int. 1
[2] https://creodias.eu/. 1
[3] https://sobloo.eu/. 1
[4] https://mundiwebservices.com/. 1
[5] https://www.wekeo.eu/. 1
[6] https://www.onda-dias.eu/cms/. 1
[7] https://sentinels.space.noa.gr/. 1
[8] https://copernicus.nci.org.au/sara.client/#/home. 1
[9] https://finhub.nsdc.fmi.fi/#/home. 1
[10] https://data.sentinel.zamg.ac.at/dhus/#/home. 1
[11] https://peps.cnes.fr/rocket/#/home. 1
[12] https://code-de.org/de/. 1
[13] https://www.collgs.lu/. 1
[14] https://colhub.met.no/#/home. 1
[15] https://ipsentinel.ipma.pt/dhus/#/home. 1
[16] https://sedas.satapps.org/. 1
[17] http://swea.rymdstyrelsen.se/portal/. 1

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," *CoRR*, vol. abs/1604.01685, 2016. [Online]. Available: http://arxiv.org/abs/1604.01685 1, 3

[19] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038 1

[20] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018. 1

[21] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587 1

[22] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. [Online]. Available: http://arxiv.org/abs/1802.02611 1

[23] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6230–6239. 1

[24] D. Lin, Y. Ji, D. Lischinski, D. Cohen-Or, and H. Huang, "Multi-scale context intertwining for semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1

[25] J. Fu, J. Liu, Y. Wang, and H. Lu, "Stacked deconvolutional network for semantic segmentation," *CoRR*, vol. abs/1708.04943, 2017. [Online]. Available: http://arxiv.org/abs/1708.04943 1

[26] H. Ding, X. Jiang, B. Shuai, A. Q. Liu, and G. Wang, "Context contrasted feature and gated multi-scale aggregation for scene segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2393–2402. 1

[27] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[28] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988. 1

[29] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8edd72158ccd2a879f79cb2538568fdc-Paper.pdf 1

[30] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: segmenting objects by locations," *CoRR*, vol. abs/1912.04488, 2019. [Online]. Available: http://arxiv.org/abs/1912.04488 1

[31] B. Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," 08 2017. 1

[32] N. Gao, Y. Shan, Y. Wang, X. Zhao, Y. Yu, M. Yang, and K. Huang, "SSAP: single-shot instance segmentation with affinity pyramid," *CoRR*, vol. abs/1909.01616, 2019. [Online]. Available: http://arxiv.org/abs/1909.01616 1

[33] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768. 1

[34] S. Liu, J. Jia, S. Fidler, and R. Urtasun, "Sgn: Sequential grouping networks for instance segmentation," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3516–3524. 1

[35] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar, "Panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[36] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5454–5463. 1

[37] Q. Pham, D. T. Nguyen, B. Hua, G. Roig, and S. Yeung, "JSIS3D: joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields," *CoRR*, vol. abs/1904.00699, 2019. [Online]. Available: http://arxiv.org/abs/1904.00699 1

[38] M. Tarasiou, R. A. Güler, and S. Zafeiriou, "Context-self contrastive pretraining for crop type semantic segmentation," *CoRR*, vol. abs/2104.04310, 2021. [Online]. Available: https://arxiv.org/abs/2104.04310 2, 4

[39] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312 3

[40] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015. 3

[41] M. Rußwurm and M. Körner, "Multi-temporal land cover classification with sequential recurrent encoders," *ISPRS International Journal of Geo-Information*, vol. 7, no. 4, p. 129, Mar 2018. [Online]. Available: http://dx.doi.org/10.3390/ijgi7040129 4

[42] R. Rustowicz, R. Cheong, L. Wang, S. Ermon, M. Burke, and D. B. Lobell, "Semantic segmentation of crop type in africa: A novel dataset and analysis of deep learning methods," in *CVPR Workshops*, 2019. 4