IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 January 2024, accepted 10 March 2024, date of publication 18 March 2024, date of current version 27 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3379142

APPLIED RESEARCH

Hyperparameter Optimization for Large-Scale Remote Sensing Image Analysis Tasks: A Case Study Based on Permafrost Landform Detection Using Deep Learning

AMAL S. PERERA[1], (Member, IEEE), CHANDI WITHARANA[1], ELIAS MANOS[1], AND ANNA K. LILJEDAHL[2]

[1]Department of Natural Resources and the Environment, University of Connecticut, Storrs, CT 06269, USA
[2]Woodwell Climate Research Center, Falmouth, MA 02540, USA

Corresponding author: Amal S. Perera (amal.perera@uconn.edu)

This work was supported in part by U.S. National Science Foundation under Grant 1720875, Grant 1722572, Grant 1927872, Grant 1927723, and Grant 1927729; in part by the Supercomputing Resources were provided by the eXtreme Science and Engineering Discovery Environment under Award DPP190001; in part by Texas Advanced Computing Center under Award DPP20001; and in part by the Polar Geospatial Center under NSF-Office of Polar Programs (OPP) under Award 1043681 and Award 1559691.

ABSTRACT Climate change pressure on the Arctic permafrost is rising alarmingly, creating a decisive need to produce Pan-Arctic scale permafrost landform and thaw disturbance information from remote sensing (RS) data. Very high spatial resolution (VHSR) satellite images can be utilized to detect ice-wedge polygons (IWPs) – the most important and widespread landform in the Arctic tundra region - across the Arctic without compromising spatial details. Automated analysis of peta-byte scale VHSR imagery covering millions of square kilometers is a computationally challenging task. Traditional semantic segmentation requires the use of task specific feature extraction with conventional classification techniques. Semantic complexity of VHSR images coupled with landscape heterogeneity makes it difficult to use conventional classification approaches to produce Pan-Arctic scale geospatial products. This leads to adapting deep convolutional neural network (DLCNN) approaches that have excelled in computer vision (CV) applications. Transitioning domains from everyday image understanding to remote sensing image analysis is challenging. This study aims to systematically investigate two main obstacles confronted when adapting DLCNNs in large-scale RS image analysis tasks; 1) the limited availability labeled data sets and 2) the prohibitive nature of hyperparameter tunning when designing DLCNNs that can capture the rich characteristics embedded in remotely-sensed images. With a case study on the production of the first pan-Arctic ice-wedge polygon map using thousands of VHSR images, we demonstrate the use of transfer learning and the impact of hyperparameter tuning with a 16% improvement of the Mean Average Precision (mAP_{50}).

INDEX TERMS Remote sensing, deep learning, hyperparameter optimization, terrain mapping, convolutional neural networks, climate change, environmental monitoring, Arctic tundra, mask R-CNN.

I. INTRODUCTION

Over the last two decades, there has been an upsurge of very high spatial resolution (VHSR) satellite image acquisitions in the Arctic region. The arrival of commercial satellite sensor platforms, such as Maxar and Planet, has mainly propelled this growth. VHSR image data archives are now at the petabyte scale and will soon be turning to exabyte scale. A major Maxar imagery repository hosted by Polar Geospatial Center (PGC), which is freely accessible to Arctic

The associate editor coordinating the review of this manuscript and approving it for publication was Geng-Ming Jiang.

researchers funded by the U.S. National Science Foundation (NSF), offers transformational opportunities to monitor environmental changes happening in the Arctic region, where field observations are sparse both spatially and temporally [1], [2].

The Arctic permafrost region is a unique landscape composed of Earth material that remains below 0°C for at least two consecutive years and occupies approximately 24% of the exposed landform of the northern hemisphere [3]. Ice-wedge polygons (IWPs), the most widespread landform in the cold continuous permafrost regions, are an atypical surface feature that manifest the presence of subsurface wedge ice [4]. VHSR remote sensing imagery opens a new window of opportunities to map IWPs from local-, regional-, to Pan-Arctic scale [4]. It is vital to automate the process of deploying Pan-Arctic scale mapping and classification approaches [4]. This leads to the adaptation of deep learning (DL) approaches that have excelled in computer vision (CV) applications [5] for remote sensing (RS) image analysis tasks. Despite their remarkable performances in CV applications, DL approaches inherit their own challenges.

Thus, the transition from one domain – e.g., everyday image understanding - to another – e.g., RS image analysis - is not straightforward [5]. Two main obstacles in using DL methods in RS applications are; 1) the limited availability of training data sets and 2) the prohibitive nature of determining the configuration parameters for the DL network. In literature, the search for optimal hyperparameters is referred to as hyperparameter tuning and is imperative when designing DL models that can capture high-level semantics [6].

VHSR images have complex and diverse patterns non-linearly aggregated to multiple spatial scales that DL algorithms can exploit. Often large learning models (LLMs) are required to exploit such complex patterns. Due to the computational cost associated with training these LLMs, iteratively determining the model configuration parameters for RS applications is a challenging task. As such, practitioners in most cases tend to use default model parameters. The case of reusing default or slightly modified parameters on a DL model could equate to asking a particular individual to identify objects on an image using a pair of eyeglasses prescribed for another individual. The primary objective of this work is to demonstrate the impact of hyperparameter tuning on the model outcome and to identify a systematic process in a generic form that can be used to engage in hyperparameter tuning based on the available computational resources for DL- RS applications.

In section II, background to the problem of using computer vision for remote sensing, the success of CNNs, and the lack of hyperparameter optimization are presented. In section III, the case study of mapping IWPs, experimental setup and the approach taken to optimize hyperparameters are introduced. In section IV, hyperparameter optimization results are compared and discussed, followed by the conclusion in section V.

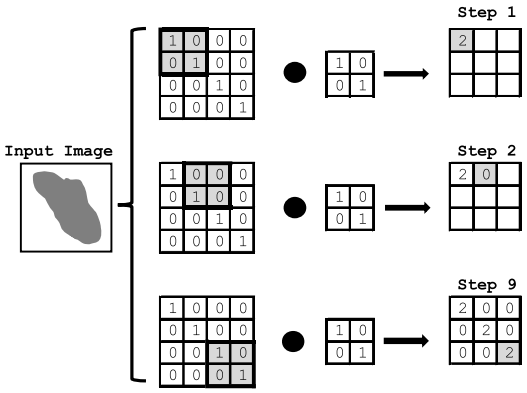


FIGURE 1. Convolution operation example for the first two steps and the last step of a convolution operation, illustrating how the resulting output highlights the shape feature in the original input in an aggregated compressed representation.

II. BACKGROUND

A. SUCCESS IN COMPUTER VISION

The performance of conventional machine learning (ML) in general and CV in particular depend on the choice of data representation [7], [8], [9], [10], [11]. In most traditional approaches, effort is largely spent on preprocessing pipelines and data transformation to enable better representations so that ML can succeed [10], [11]. The use of DL on the MNIST (Modified National Institute of Standards and Technology) optical character recognition image data set [10] to break away from the best performing support vector machine (SVM) based approach is a pioneering effort in CV. Higher performance of DL over SVM shows the representational learning limitations in using traditional machine learning for image processing. Simple CV models are bound to fail as they rely on the smoothness of the function to be learned and may not be able to capture a sufficient quantum of the complexity of interest. Any attempts to provide a suitable feature space would run into the curse of dimensionality [10], [11]. DL's success in image understanding, beyond human capabilities in some cases, clearly shows the ability of DL techniques to capture and represent the inherent complexities in images. Multiple CV applications using auxiliary data representation techniques, such as histogram of oriented gradients (HOG) [12], scale invariant feature transformation (SIFT) [13], and bag of words (BoW) [14] have shown improved performance, illustrating the limitations in conventional ML for higher dimensional data understanding.

B. REMOTE SENSING WITH DEEP LEARNING

DL addresses the data representation issues in traditional machine learning based image processing approaches [10], [11]. In DL, feature extraction is automated by eliminating the requirement to tweak the data representation based on domain knowledge [11]. DL models use a multi-layered approach to extract higher level discriminative features. Initial layers capture the low-level features and the final layers capture high

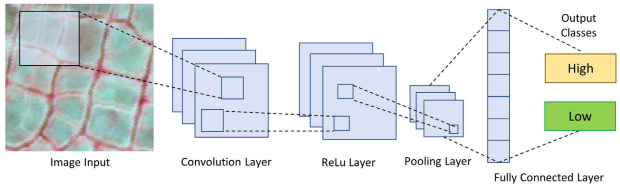


FIGURE 2. Basic CNN Architecture [11] The output of the convolution layer is passed through the rectified linear unit (ReLu), where the non-linearity of the image features extracted by the convolution layer is further enhanced.

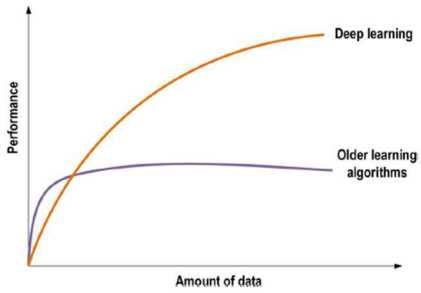


FIGURE 3. Performance of DL with training data size [11] represented by a hypothetical graph indicating the relationship between the training data size and the final performance of a ML model.

level features which enable classification [11]. The convolutional neural network (CNN) in DL enables discriminative image features to be captured efficiently, aided by a collection of kernel-based convolutional filters. The N-dimensional matrix of the input image is convolved with kernel-based filters to extract discriminative features. A kernel is a grid of trainable weight values which are randomly assigned initially and adjusted in training. The kernel window slides over while computing the dot product with the input at each step capturing the features in the input image in an aggregated form. Figure 1 is an example of feature extraction.

Figure 2 shows a basic CNN architecture. The pooling layer in a CNN enables the subsampling of the feature maps generated by the convolution layers. This reduces the dimensionality of large feature maps but retains the dominant features. Activation layers control information flow enabling the capture of image features. The final layer of a CNN is the fully connected layer that outputs the decision based on the captured features from the previous layers. During training with multiple images, the weights and biases in the model are adjusted based on the loss, which is computed by comparing the expected output against the actual output at each iteration until the loss reaches an acceptable level.

C. CHALLENGES IN DEEP LEARNING

A downside of DL is that large collections (14 million images in the case of ImageNet [15], 200,000 for MaskR-CNN [16]) of training samples are required for a model to learn sufficient information to execute inferencing with high quality [11], [17]. Figure 3 compares training dataset size for traditional ML with DL. A large training set is used by DL models to capture all possible discriminating characteristics [17], [11]. In DL-based RS, it is a difficult task to manually annotate the data to create a large training data set. Transfer learning overcomes this challenge [17], [11] by starting with a well-trained model, where the weights are already trained for a similar task, and concludes by only training the last few layers of the DL model with the task specific training data. The initial layers, which are responsible for feature extraction in such pre-trained models, are typically frozen during training in order to leverage the weights already learned.

Another challenge in DL is hyperparameter tuning. Due to the complexity and configuration variability of large DL models, we must use the best-performing configuration based on hyperparameter tuning to obtain the optimal inference results from the model [6], [18]. In some cases, the hyperparameter space can be prohibitively large, and the requirement to iterate over possible training configurations to arrive at the optimum using large RS images can be a serious computational challenge.

D. HYPERPARAMETER TUNING

From the initial introduction of neural networks, one of the critical drawbacks was hyperparameter tuning. In an initial publication done in 1967 [19], the issue is described as the problem of "learning the learning rule." This is compounded further by the fact that NNs were black boxes that produced results with very minimal model explainability to draw intuitions on how well it is performing. NN based DL performance depends heavily on the correct settings of many internal parameters [20], [21], [22], [23]. The best performing models most often turn out to be larger networks with multiple parameters that need to be set. This makes it desirable for both the researchers and practitioners to set these hyperparameters automatically or semi-automatically [22], [24]. Some of the widely adapted hyperparameter tuning approaches include, Grid Search, Random Search, Bayesian optimization, and Hyperband optimization. In Grid Search, all possible combinations of different hyperparameters are evaluated, and in the case of DL-based RS it is prohibitively expensive. Random search attempts to reduce the cost of grid search by limiting the search to a set of randomly selected hyperparameters from the exhaustive list, hoping that one of them will lead to some form of optimal solution without any guarantees. Bayesian optimization uses a probabilistic model to select the next set of hyperparameters to evaluate based on the outcome of executing the training. Hyperband uses a multi-armed bandit strategy for hyperparameter optimization that takes advantage of reducing the allocated resources for a trial in the search, based on considering the estimated return on (resource) investment (RoI) for the trials.

E. HYPERPARAMETER TUNING IN DL BASED RSI ANALYSIS

Any work that uses DL (specifically for RS) must explore and discuss hyperparameter optimization, unless the practitioners are using a model previously used for a similar task that has

TABLE 1. Summary of hyperparameter tuning on deep learning based remote sensing literature 2015 to 2022.

Category	Count	References
Not mentioned	13	[25]–[37]
Mentioned tuning with no reporting	1	[38]
Mentioned tuning with minimal reporting	4	[39]–[42]
Mentioned tuning with reproducible reporting	6	[43]–[48]

significantly explored the respective hyperparameter space. Automated hyperparameter tuning is particularly important in DL RS applications as it benefits in multiple ways, such as: the ability to obtain strong anytime performance, the availability and effective use of parallel resources for training, the reduction of the dependence on ML expertise, and the avoidance of trial and error approaches to obtain the best performing model [22]. In DL RS, data preprocessing, feature engineering, model selection, and model parameter selection are some of the key design decisions that need to be made to deploy a DL-based RS product. With the advances in the field, the availability of multiple options for each step in the pipeline creates a vital and crucial need for hyperparameter optimization.

We conducted a meta-search of 24 recent works in DL-based RS to explore the use or non-use of hyperparameters (Table 1, See Appendix for further discussion) Most of the work concludes model training as soon as the model performance reaches results that are better than the reported results and the choice of model parameters are accepted without exploring any further. This could be due to the high computational cost inherent in tuning. However, with the availability of parallel computing environments makes hyperparameter tuning feasible and should not be only considered as future work by researchers.

Although it is not exhaustive, our short literature survey reflected that most of the work done in RS using DL does not discuss the importance of hyperparameter tuning. Since the cost of hyperparameter tuning is high due to the complexity of the DL model and the size of training data, it is essential to discuss the cost of tuning and also methods and means, such as parallel computing, that can be utilized for hyperparameter tuning. For example, the learning rate is one of the key elements of a DL setup that affect the outcome of the models. It is obvious that most of the work published in DL-based RSI analysis would execute a few random trials with multiple learning rates and use the best value. A systematic approach to learning rate optimization should be conducted and reported as a bare minimum.

III. METHODS

A. CASE STUDY

To demonstrate the quality and cost impact of hyperparameter tuning on DL-based RSI analysis in a computational cost-constrained environment, we select an Arctic permafrost science use case, that utilizes VHSR satellite image scenes acquired by the Maxar sensors (e.g., Quickbird, Worldview-02) at a spatial resolution of 0.5 m with a typical footprint size of 20 km × 20 km (∼160 million pixels per image scene) [48].

Our aim is to train a DL model that can semantically segment IWPs over a large spatial domain using VHSR data. It is estimated that two-thirds or more of the Arctic is occupied by polygonized ground, but the exact extent and the prevailing IWP types (i.e., whether the ice-wedges experienced thaw or not) are largely unknown [49], [50]. Degradation of IWPs is known to occur over a short period of time compared to aggradation due to the accumulation of organic and mineral soil above the ice-wedges [51]. Analyzing the spatiotemporal dynamics behind the ice-wedge polygonal tundra demands the production of geospatial maps documenting prevailing IWPs with type classifications. The Arctic science community has a limited grasp of the Pan-Arctic scale spatiotemporal dynamics of IWPs despite the alarming signals of changing climate. These IWPs have multiple size configurations and spectral characteristics, making them heterogeneous data sets with numerous inference challenges. Another challenge for this data set is in creating a training data set through manual annotation by going through a selected set of tiles sampled from the satellite images and marking the IWPs. All these are typical challenges faced in DL-based RSI analysis.

B. MASK-RCNN MODEL TRAINING

Since the goal is to segment and classify into multiple instances of IWPs (i.e. low and high), we utilized Mask RCNN, which is a semantic object instance segmentation model [16]. Figure 4 illustrates the Mask RCNN [16] architecture. Mask RCNN is used in RS DL due to its success in the field of CV [4], [52]. The requirement to configure multiple components in the Mask RCNN architecture makes hyperparameter tuning vital to have highly accurate outputs.

Operational model inferencing spans over a very large spatial domain (>5 million km^2) of tundra comprising heterogenous landscape characteristics. Thus it is vital to make sure that the model used is able to capture most of the landscape variability in the training data to accurately map IWPs across the Arctic. Figure 5 shows the process flow for training the model. As indicated in Figure 5, the hyperparameter tuner will iteratively suggest new model configurations based on the model performance. This process will continue until the process reaches a given stopping condition and produces the final tuned model to be assessed against the test dataset for the final model performance reporting. It should be noted that the training of the weights and biases in the DL model is separated from the tuning of the parameters of the model. Though the terms ''train'' and ''tune'' are used for these two concerns, essentially, both are doing similar tasks in yielding the best configuration for the model. It is important to separate the two, rather than do both at the same time, since there is a direct impact and conflict between the two. For example, the number of layers on a network is a hyperparameter, and if we

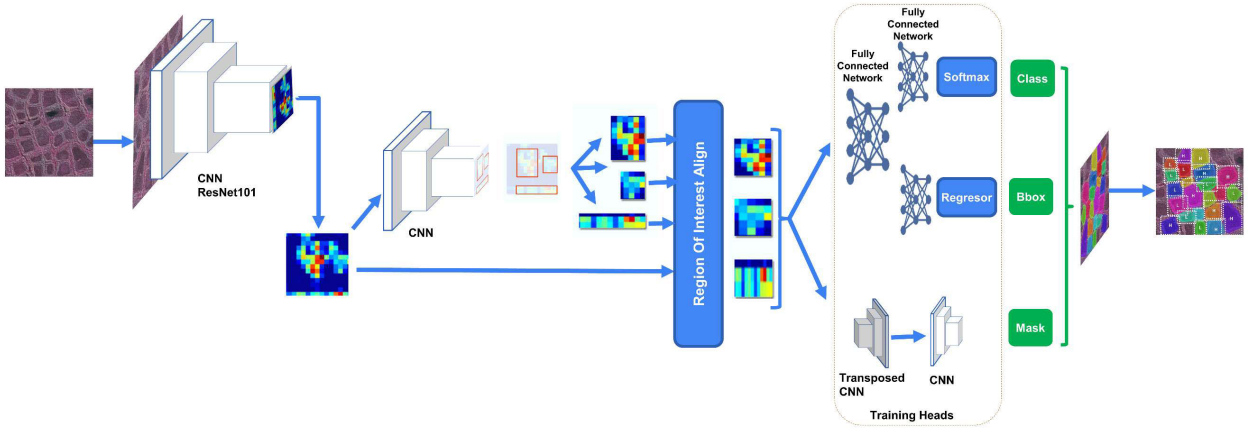


FIGURE 4. Mask RCNN model architecture [16] that involves multiple CNNs with other fully connected networks that are combined for semantic instance segmentation. The architecture contains multiple components to be configured appropriately.

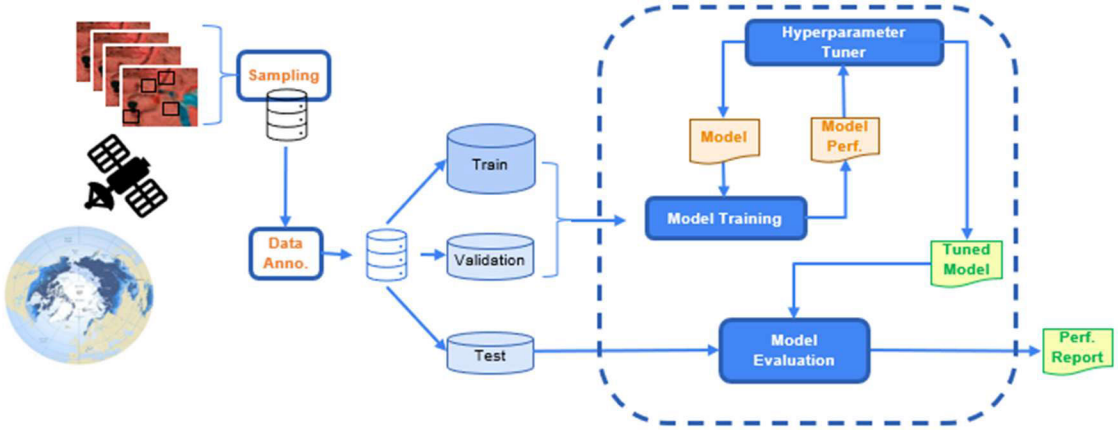


FIGURE 5. Hyperparameter tuning process will suggest model configurations based on the performance of the model.

attempt to tune it while training the network weights, there may be multiple different numbers of weights in the network at different epochs in a training session, which confuses the learning process.

C. TRAINING DATA SIZE LIMITATIONS

For this science use case, the training data set was sampled from a few representative satellite image scenes. Table 2 shows the distribution of the training samples across the train, validation, and test set that includes almost 25,000 IWPs on 670 image tiles. In each image tile, we manually digitized the outline of IWPs and gave the class label of ''high'' or ''low'' based on their morphometry, [48]. This training data will enable a model to segment boundary, and classify as low- or high-centered IWPs.

In this case study, transfer learning strategy is used to address the limited training data issue. Transfer learning is adopted with the use of ResNet101, a pre-trained model already trained using an extensive collection of images as a component of the Mask R-CNN. Model weights are initialized using a set of weights to detect the Balloon data set [53], and the layers trained are limited to layer heads.

TABLE 2. Training data set distribution.

	Low	High	Total	Tiles
Train	11131	6404	17535	487
Validation	2104	1584	3688	103
Test	2727	1532	4259	80
Total	15962	9520	25482	670

D. COMPUTE ENVIRONMENT

The computing infrastructure used for this case study is from the U.S. NSF-funded Texas Advanced Computing Center (TACC) Peta-scale HPC Frontera [54]. Frontera has 90 GPU nodes with 4 NVIDIA Quadro RTX 5000 GPUs per node that can be utilized for training and inferencing DL based RS. This enables a DL-based RS practitioner to execute multiple training iterations in a distributed mode in order to exploit the hyperparameter space. Although not described in this work, TACC resources were also utilized to map 5 million km^2 of land area into IWPs [2]. In this work, we demonstrate the utilization of TACC infrastructure and the supporting computing environment to execute hyperparameter tuning for our case study in DL RS.

E. HYPERPARAMETER SEARCH SPACE

For the case study, we used the Mask RCNN with ResNet101 backbone [16]. The publicly-available Mask RCNN model codebase has about 50 configurable parameters that the user can change. Considering five steps per parameter, a rough estimation of the search space can be made after removing the binary parameters and some parameters that are selected based on the nature and configuration of the data set. Size of the search space, after selecting 20 parameters to train, would be 5^{20}. Considering executing 200 epochs per trial, utilizing 4 GPUs per node on Frontera HPC system on TACC with a batch size of 16, costing approximately 45s per epoch, a single experiment would require approximately 2.6×5^{20} compute hours. Though using all 80 nodes is not feasible on TACC Frontera, considering the shared nature of the resource, using all 80 nodes would still require a significant amount of time. Some of the parameters can be left at the default or widely used values for similar applications based on published literature. An example of a set of selected parameters and possible range is given in Table 3. Note that the objective of the case study is to show the selected hyperparameters' impact on the model's final outcome and to motivate the applicability of hyperparameter tuning to build better DL models. The restricted grid search for 7 parameters would still require 200,000 hours of computing time. On TACC Frontera, this would cost 600,000 Service units (SU), as the GPU nodes cost three times the cost of a single node due to the added benefit of being able to use the GPU's in parallel. And in this case, with the use of Mask RCNN, we make maximum use of the GPUs by processing multiple image tiles at the same time during training.

In addition to these configuration parameters, there are other parameters that are not easily configurable through the configuration file with the publicly available Mask RCNN codebase, but can be configured by tweaking the code since most of the libraries used in the code make it possible to change these configurations and the respective parameters. For example, the learning rate can be scheduled instead of having a fixed value. Rather than using the standard Stochastic Gradient Descent (SGD) for learning, a slightly sophisticated Adam optimizer can be used with its' own parameters. Such parameter options make the problem of hyperparameter optimization even more challenging. Still, they should not be an excuse for not exploring the parameters as is done in most of the studies. A very promising DL-based RS product may be dropped due to non-exploration of the hyperparameter space.

F. PROPOSED SEARCH ALGORITHM

In this work, we propose Algorithm 1 given below, which can be categorized as a greedy algorithm as a possible approach to explore the hyperparameters with an available computing budget to arrive at a suboptimal solution.

In our algorithm, the selected parameters are explored one at a time over a selected range at a given step size. Space

TABLE 3. Hyperparameter space for case study.

Hyperparameter	Description / Impact	Range
rpn_nms _threshold	Non-max suppression threshold to filter RPN proposals.	0.4 – 0.8
rpn_class_loss	Region proposal network class loss	0.5 – 1.5
rpn_bbox_loss	Region proposal network bounding box loss	
mrcnn_class_loss	Mask RCNN class loss	
mrcnn_bbox_loss	Mask RCNN bounding box loss	
mrcnn_mask_loss	Mask RCNN mask loss	
detection_min_ confidence	Minimum confidence to consider a detection as a true detection.	0.2 – 0.6

Algorithm 1 Search for Best Config

input:
RnkPrmLst,RnkPrmLstRng$_{\min}$,RnkPrmLstRng$_{\max}$,stps
output: cfg*

```
cfg* ← {all params set to mid point of param range}
bestVal ← MaxLoss
for prm ∈ RnkPrmLst do
    prmRng_min ←RnkPrmLstRng[prm]_min
    prmRng_max ←RnkPrmLstRng[prm]_max
    for prmVal ∈ {x|PrmRng_min ≤ x ≤ PrmRng_max,
     in stps} do
        newCfg ← {cfg* ∩ (prm,prmVal)}
        ValLoss ← TRAIN(newCfg)
        if BestVal > ValLoss
            cfg* ← newCfg
            bestVal ← valLoss
```

Algorithm 1 Hyperparameter Optimization Algorithm in Pseudocode:RnkPrmLst is the selected list of ranked parameters to be optimized, RnkPrmLstRng indicates the min and max values for the respective Parameter, stps are the number of steps to be checked, and cfg is the hyperparameter configuration.

complexity is constant and the time complexity is linear time as indicated in expression (1) where TC: Time Complexity, SBC: (Proposed)Search Best Configuration Algorithm, BF: Brute Force Algorithm. Compared to the computational cost of a brute force search (2) the proposed algorithm has a major reduction in cost due to the assumption of having no interdependency among parameters, which may not hold true in most cases. This loss of accuracy due to the interdependency of parameters can be mitigated marginally by using a greedy approach by ranking the selected parameters based on a general understanding of DL, and any other reported similar work as suggested in this work.

$$\text{TC (SBC)} = O\left(SelectedParams \times range\_steps\right) \quad (1)$$

$$\text{TC (BF)} = O\left(SelectedParams^{range\_steps}\right) \quad (2)$$

For example, based on general understanding of DL, learning rate can be considered as one of the topmost parameters

that affect the outcome [55] and can be ranked higher for tuning. In CV tasks, loss function is also another aspect that impacts the outcome [56]. The loss function plays a critical role in providing feedback for the learning process. With an understanding of the DL model used and the domain where it is applied, the loss function can also be given high priority in tuning. In the case of the Mask RCNN model, used in this work the final loss function is an aggregation of multiple loss functions based on the performance of the final mask, bounding box, class, and the region proposal network class and bounding box loss. The model uses a weighted aggregate, and the weights can be tuned. Within the collection of weights, a higher priority can be given to the region proposal network loss weights based on an understanding of the Mask RCNN model as it feeds into the final layer of training heads. In this work, the parameters were ranked for the search algorithm as follows; learning_rate, rpn_class_loss, rpn_bbox_loss, mrcnn_class_loss, mrcnn_bbox_loss, mrcnn_mask_loss, rpn_nms_threshold, detection_min_confidence.

In our approach, each parameter in the ranked list is taken one at a time and tested within the preselected range. The range step that gives the best validation performance is locked into the best configuration before testing the next parameter in the ranked list. Initially, the configuration is set to the mid values of all the parameters. In this algorithm, the step size is a constant for all parameters in order to make it easier to present, but the algorithm can be easily modified to take a respective step size for each parameter. As can be seen, it is a straightforward algorithm that can be easily deployed using any scripting language, provided the DL codebase used has an interface to modify the parameters. It can also be done manually by running each training algorithm by setting the values in the configuration file of the DL model. This algorithm can be categorized as an embarrassingly parallel algorithm, as the inner loop of this algorithm can be executed in parallel on multiple nodes on an HPC as the individual training steps do not have any dependency on each other. In the case of parallelization, the results of each of the sessions need to be reported to identify the best configuration value to proceed to the next iteration on the outer loop. With parallelization, the time required for tuning can be reduced by a factor of the number of range steps, provided the inter-node communication to initiate and return the final result is negligible compared to a single training run.

Using the suggested algorithm for 10 range steps on 10 parameters will require 200 compute hours on TACC Frontera, costing only 600 SU's. Distributing it over the 10 nodes will only require 20h of wait time to obtain the results. Also, the reported loss values of the search space executed can be analyzed to make sure that the values selected are appropriate. In the next few subsections, we discuss some of the hyperparameters explored to arrive at the best possible solution within the available compute budget for the selected case example. The objective of the discussion is to highlight the impact of each parameter.

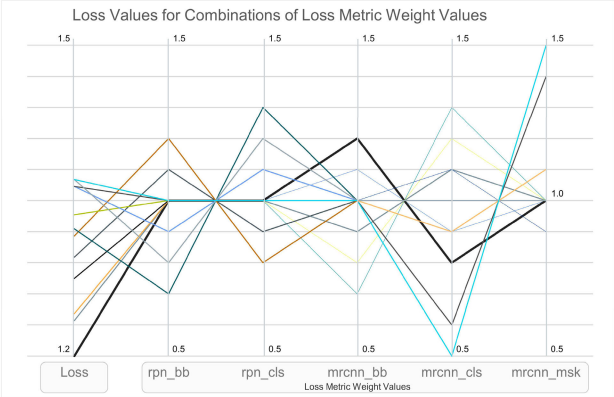


FIGURE 6. High-dimensional trial plot of loss weight values indicating multiple combinations giving similar results.

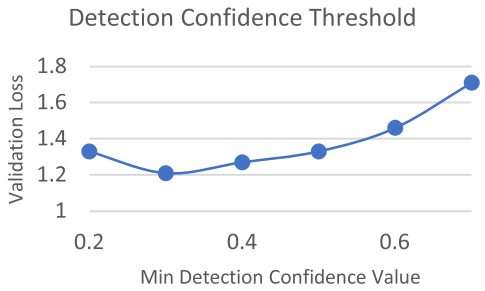


FIGURE 7. Detection Confidence Threshold: the change in validation loss for different detection confidence thresholds.

G. HYPERPARAMETER TUNING.

1) LOSS FUNCTION WEIGHT VALUES

The following section presents the hyperparameter tuning executed and the outcomes. One of the crucial aspects of training a ML model is to use the most suitable objective function for the application. The objective function guides the training process by providing the correct feedback to the learning algorithm in order to adjust the weights and biases in the neural network. In general, the objective function's form and parameter values can be adjusted to suit the application and the data set. In the case of the Mask RCNN model used in the case study, the proposing of possible regions for objects, the selection of the bounding boxes, classification and mask generation contribute to the final outcome. The loss function is computed using the individual loss values from all these components. Rather than using the default equal flat value, the weights of the individual components can be adjusted to observe the impact of the weight values on the final outcome. Figure 6 shows a high dimensional trial plot for the loss weight values. The loss weight values were configured to be between 0.6 and 1.6, and the sum of all weights to be 5 so that the validation loss does not get affected by the respective weight value at a given experiment. It can be clearly observed that the best possible loss value and similar loss values can be achieved with multiple combinations of weight values. Although the graph may look like an exhaustive grid search, an exhaustive grid search was not carried out. As indicated

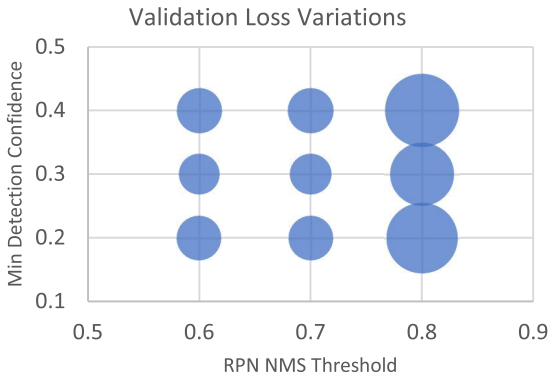


FIGURE 8. Validation loss (size of circle) variations for Min Detection threshold and RPN NMS Threshold.

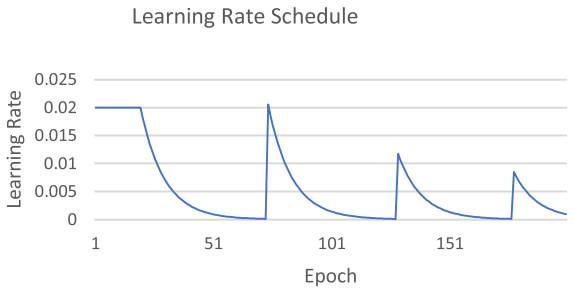


FIGURE 9. Learning rate schedule with multiple peaks that decay.

previously, a simple sequential search in a nested loop is done on the 7 parameters, where a single parameter is checked and locked before the next parameter is tested. Unfortunately, the inter-parameter co-relations would be missed in this approach leading it to a suboptimal solution, but this approach is taken due to the prohibitive nature of the computing cost, and the purpose of this work is to show the impact of hyperparameters on the final outcome.

2) DETECTION MINIMUM CONFIDENCE AND RPN NMS THRESHOLD

After exploring the loss weights and selecting the best values for the loss weights the Minimum Detection Confidence threshold is explored to observe the impact on the model training as shown in Figure 7. The value 0.3 is clearly giving us a better validation loss compared to the other values evaluated. To further illustrate the impact, Figure 10 shows the impact of the Minimum Detection Confidence on the outcome on three different examples from the test set. When the Minimum Detection Confidence value is low, the model tends to pick up more ground truth polygons as indicated on the three images on the right side, that has more brown polygons compared to the few green IWPs that have not been identified when the detection confidence is set at 0.7 indicated on the left side of the figure. It can also be noted when the minimum confidence is set at a lower value the systems tend to pick a few extra false positive IWPs as indicated with the red colored polygons. It is vital to pick an optimum threshold for the given task and the three examples further confirm the impact of this particular hyperparameter on the final RS outcome.

The Region Proposal Network Non Maximum Suppression (RPN NMS) threshold directs the algorithm to filter out overlapping region proposals during the training process. In the NMS process, it will take all the proposed region proposals (RPs) and take the topmost proposal based on the probability of it being a RP and compare it with all the other RPs IoUs. If an IoU is above the given threshold, we remove it from the list because we obviously have a better RP to represent that particular RP. So, if we have a smaller threshold, we will be removing less RPs and checking more RPs. Checking more RPs can improve the accuracy since it can avoid missing out on possible ground truths RPs, but there is a cost tradeoff with respect to the NMS threshold. Having more region proposals can improve the accuracy, but would come with an additional computational cost of having to go through all the additional proposals. Figure 8 shows the variation of the validation loss for different values of the Minimum Detection Confidence and the RPN NMS threshold. In this case, it is clear that we are able to reduce the validation loss by decreasing the threshold from 0.8 to 0.7 and testing more region proposals, but a further decrease to 0.6 is not making a significant difference to the validation loss. So, for this case example we can set the RPN NMS to a value of 0.7. Figure 7 and Figure 8 shows the variation of the loss function for the selected set of range steps for each of the parameters selected. The number of range steps to be explored was selected based on the resources available for tuning and did not allow for exploration of the parameters at a finer granularity. The results of the 9 data points on Figure 8 shows a very small portion of the search space that requires exploring to arrive at the most suitable parameters for the model.

3) LEARNING RATE SCHEDULER

Learning Rate (LR) influences the quantitative adjustment done at each step of the learning process. A higher LR can miss the suboptimal solution, a lower LR may take far too many iterations, and the DL solution may miss reaching it due to the limits set with the maximum number of epochs. As discussed, an unsuitable learning rate could lead to a waste of computational resources. This is aggravated further in the case of RS dealing with large training data sets with limited computational budgets. There are multiple widely used LR schedulers [55] that can be is used. At a very minimum, the setting of the LR for the given application of DL should be explored. Further exploration of scheduling the LR to change based on the current epoch can be considered. The original Mask RCNN paper [16] suggests changing the LR by a factor of 10 at the 120^{th} epoch without discussing the intuition or basis behind the tuning decision. There are also adaptive LR schedulers that adjust the LR based on the current performance of the learning process. For example, one of the most widely used LRs would notice no change or a very minimal change of the loss value and adjust the LR to infuse movement of the loss value that would indicate learning.

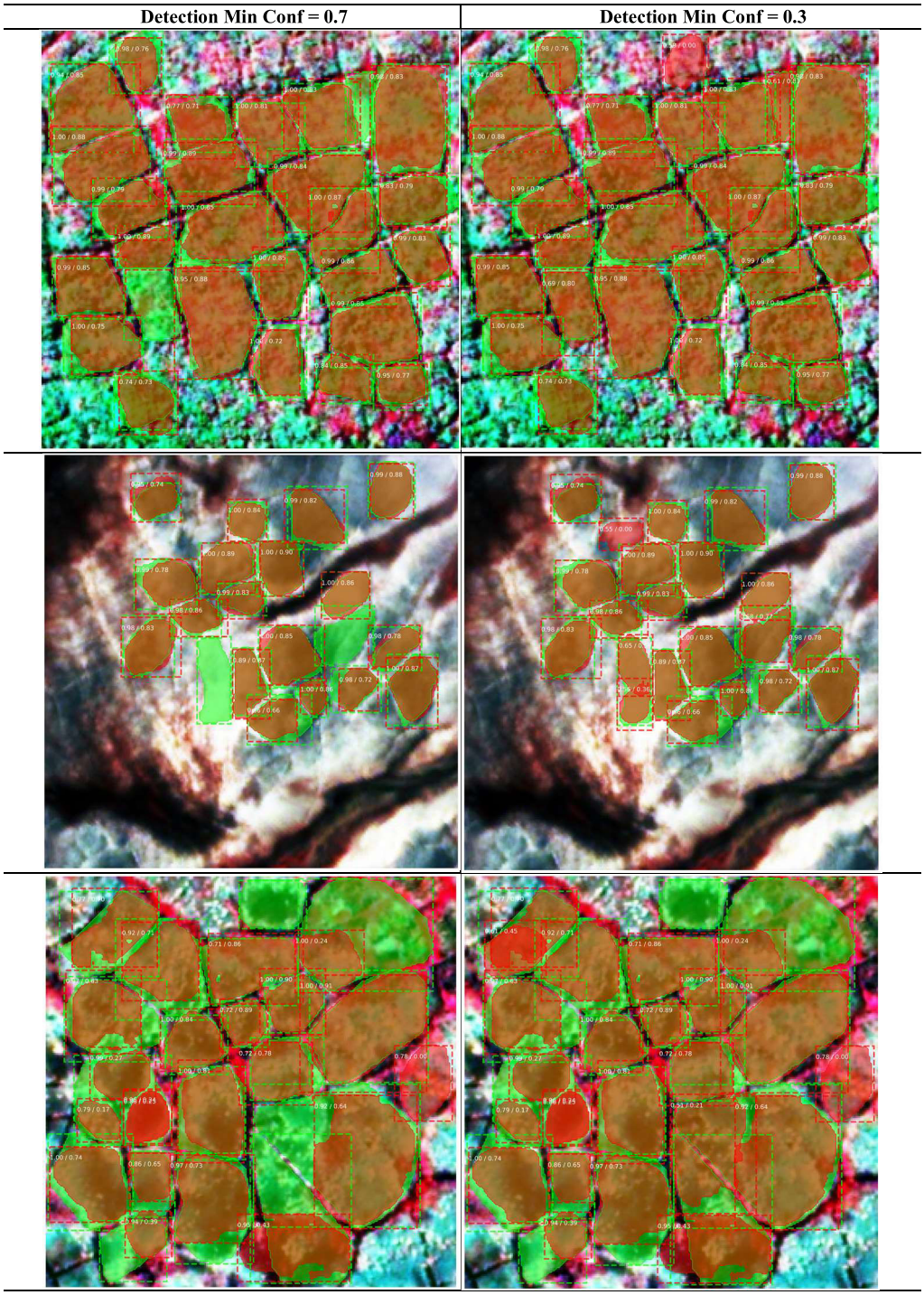


FIGURE 10. IWP Detection at two different hyperparameter values (Imagery©Maxar).

In this analysis, we have deployed a learning rate that starts at a static rate and goes on for 20 iterations and starts to decay until it reaches a threshold and would be pushed back up once again to a higher LR that would not be beyond the initial set value. The pushing to a higher value is also controlled to gradually decay to a lower level based on the epoch. The LR scheduler used is similar to the use of a sawtooth-like function that allows for learning rate annealing [57], a commonly used approach in DL. In this approach, the learning rate is scheduled to gradually decay with multiple restarts that gradually decay over time. This would add a few more parameters to the list of hyperparameters. In this

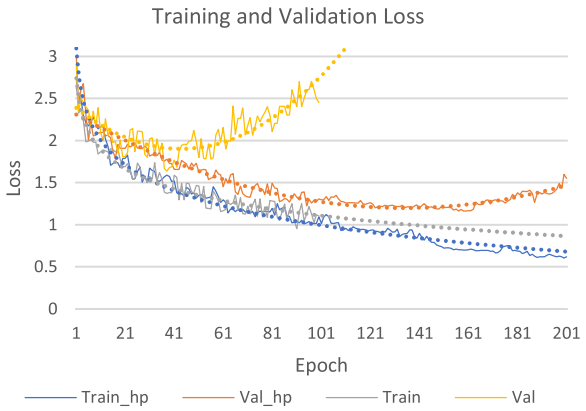


FIGURE 11. Training and Validation Loss curves where _hp indicates training with hyperparameter tuned model and the dotted lines indicate the respective trendlines.

TABLE 4. Comparison of final test accuracy with hyperparameter tuning.

	Test			Validation
	mAP_{50}	mAR_{50}	$mF1_{50}$	Loss
Without HP tuning	0.715	0.771	0.739	1.5201
With HP tuning	0.832	0.870	0.849	1.1630

case example the simple function given in Equation 1 is utilized with $\alpha = 10$, $\beta = 1.5$, $\gamma = 20$ and found to converge faster compared to non-scheduled and stepwise scheduled LRs with the standard SGD optimizer. In expression 3, the LR_i indicates the learning rate at the ith iteration and LR_0 is the initial learning rate.

$$LR_i = \begin{cases} i < \gamma, LR_0 \\ \begin{cases} LR_{i-1} < \alpha^{-4.0}, \frac{\beta}{i} \\ LR_{i-1}e^{-0.1} \end{cases} \end{cases} \quad (3)$$

Figure 9 shows the learning rate schedule used for the training based on the selected values ($\alpha = 10$, $\beta = 1.5$, $\gamma = 20$). In this work, the LR scheduler is discussed as an example of the increase in the hyperparameter space that may need to be explored when using DL approaches. Multiple trials with different learning rate schedulers would increase the resource requirement for parameter tuning.

Adam optimizer [58] is an alternative to the use of the widely used and default stochastic gradient descent approach(SGD), although there are many other variants of adaptive optimizers for learning that can be explored. In this case study, the Adam optimizer was used with default values and found not to be better than SGD for this application. The Adam optimizer's many parameters will also add a few more hyperparameters that need to be tuned in the case of using DL with Adam.

For the entire process of hyperparameter tuning that included the loss weight values, minimum detection confidence, non-maximum suppression threshold, Adam and SGD optimizers with multiple parameters, and dynamic learning rate scheduler parameters with suitable ranges can be estimated to require about 600 compute hours. This would be 1800 SU's on TACC Frontera.

TABLE 5. Confusion matrix.

		Ground Truth			
		Low	**High**	**Row Total**	**Precision (UA)**
Classification	**Low**	0.4823	0.0353	0.5177	0.9318
	High	0.0192	0.4631	0.4823	0.9601
	Column Total	0.5016	0.4984		
	Recall (PA)	0.9617	0.9291		

IV. RESULTS AND DISCUSSION

In this section, we show the final result obtained from the hyperparameter tuning analysis explained in section III-G. The comparison is done in contrast to using the model without any hyperparameter tuning, using the default configuration and some basic settings that can be inferred from the data set size and the nature of the application. Figure 11 shows the training and validation loss across 200 epochs for the model with hyperparameter tuning and 100 epochs for the model without hyperparameter tuning. The model with no hyperparameter tuning is restricted to 100 epochs since the validation loss starts to deviate around the 40^{th} epoch and does not improve after that. The deviation of the validation curve indicates a possible overfitting of the model when trained without hyperparameter tuning. From this graph, it can be observed that the training can benefit and the validation loss can be further improved from hyperparameter optimization based on the selected parameters discussed in the previous section and the respective tuning carried out for this case example.

Table 4 compares model accuracies on the test set before and after hyperparameter tuning. The results are given in the widely accepted COCO metrics for computer vision, also used in the original Mask RCNN paper [16]. With the hyperparameter tuning discussed in the previous section, the validation loss improved from 1.52 to 1.16, significantly improving the test accuracies. The mAP_{50}, mAR_{50}, and the $mF1_{50}$ all show a significant inferencing improvement for the model. An RS product produced based on the two different models would have a significant difference in the output, confirming the importance of parameter tuning in DL. It should be noted that the test data set was not used for any hyperparameter tuning, as is the case in standard DL practice, to obtain a completely independent evaluation of the model to ascertain how it would work when deployed on an entirely new data set, such as the 5 million km^2 of land area to be covered in this application.

A. CLASSIFICATION ACCURACY

The confusion matrix for the classification is given in Table 5. This confusion matrix is derived by ignoring the background as reported in most cases for instance segmentation-based RS object detection. Table 6 shows the confusion matrix with

TABLE 6. Confusion matrix with background.

		Ground Truth				
		Low	**High**	**Background**	**Row Total**	**Precision (UA)**
Classification	**Low**	0.3430	0.0251	0.0696	0.4377	0.7836
	High	0.0137	0.3293	0.0769	0.4199	0.7843
	Background	0.0696	0.0728	0	0.1424	
	Column Total	0.4263	0.4272	0.1465		
	Recall (PA)	0.8046	0.7708			

TABLE 7. Confusion matrix with background for DL model without hyperparameter tuning.

		Ground Truth				
		Low	**High**	**Background**	**Row Total**	**Precision (UA)**
Classification	**Low**	0.2788	0.0510	0.1370	0.4668	0.7313
	High	0.0327	0.1944	0.1141	0.3412	0.6825
	Background	0.0741	0.1179	0	0.1920	
	Column Total	0.3856	0.3633	0.2511		
	Recall (PA)	0.5181	0.7067			

the background as a class showing the fraction of objects identified as a particular object, though it is actually background based on the ground truth. In the computer vision literature related to instance segmentation-based object detection, background is indicated as a class with a value of 0, since there is no background class in the tabulated ground truth. From a DL point of view, the effect of the model's misclassifications predicting background areas as a particular class should be considered when making adjustments and improvements to the DL model. The mAP_{50}, mAR_{50}, and $mF1_{50}$ values indicated in Table 4 comparing the use of hyperparameter tuning include the impact of misclassification due to the background. Table 7 shows the confusion matrix for the DL model without any hyperparameter tuning. The confusion matrix allows the practitioner to identify the mistakes and the type of mistakes made by the classification model. In this case, it can be clearly observed that many background locations are misclassified as IWPs when using a model with minimal or no hyperparameter tuning. In this case, the resulting map product will have multiple false positives, thus reducing the thematic accuracy of the final output. The DL based semantic segmentation model used in this work clearly shows the impact of hyperparameter tuning.

Examples of inferencing outputs are given in Figure 12 and shows a snapshot of the final product produced by the inferencing model. Identification and classification of the IWPs to "high" and "low" can be observed. These samples are from outside the training data set and shows how it will work across the entire area that needs to be mapped.

In RS, DL models are widely used to scale up imagery-based mapping applications. DL with transfer learning provides an opportunity to train a model on a limited but representative sample of training data and use the model to infer the ground truth from large collections of high-resolution images. One of the inherent issues with DL models is the dependence of DL on multiple parameters which could affect the final outcome of the RS product inferred from the DL model. A very high-level meta search of the literature in RS DL reveals that most of the work does not indicate the use of hyperparameter tuning of the DL model. And also, the ones that reveal some form of hyperparameter tuning report the selected parameters but do not report the tuning process in detail for reproducibility. Hyperparameter tuning requires the training of the DL model multiple times, requiring a large computing budget. In the case of RS, due to the scale of the input data, this could lead to ignoring the hyperparameter tuning or engaging in very minimal effort until a reasonable solution is obtained to concentrate more on the inferencing and producing the final RS product using the available compute resources.

The primary objective of the work is to show the importance of hyperparameter tuning for DL-based RS products. The total cost of executing a grid search for hyperparameter tuning for this case study with some reductions would still be prohibitively expensive. In this case study, we propose a pseudo breadth-first search on hyperparameter tuning where we check a selected parameter, lock the best value, and explore the following parameter. To demonstrate the impact of hyperparameters, a few examples of the process carried out with results are discussed in section III-G under methods. The improvement with hyperparameter tuning can be observed in the comparison results with a model with no hyperparameter tuning. The RoI from consuming 600 compute hours or 1800 SU's can be justified considering the quality improvement.

B. RECOMMENDATIONS

Based on the findings of this case study, hyperparameter tuning can be strongly recommended before deploying DL based RS products for inferencing at operational scale. The

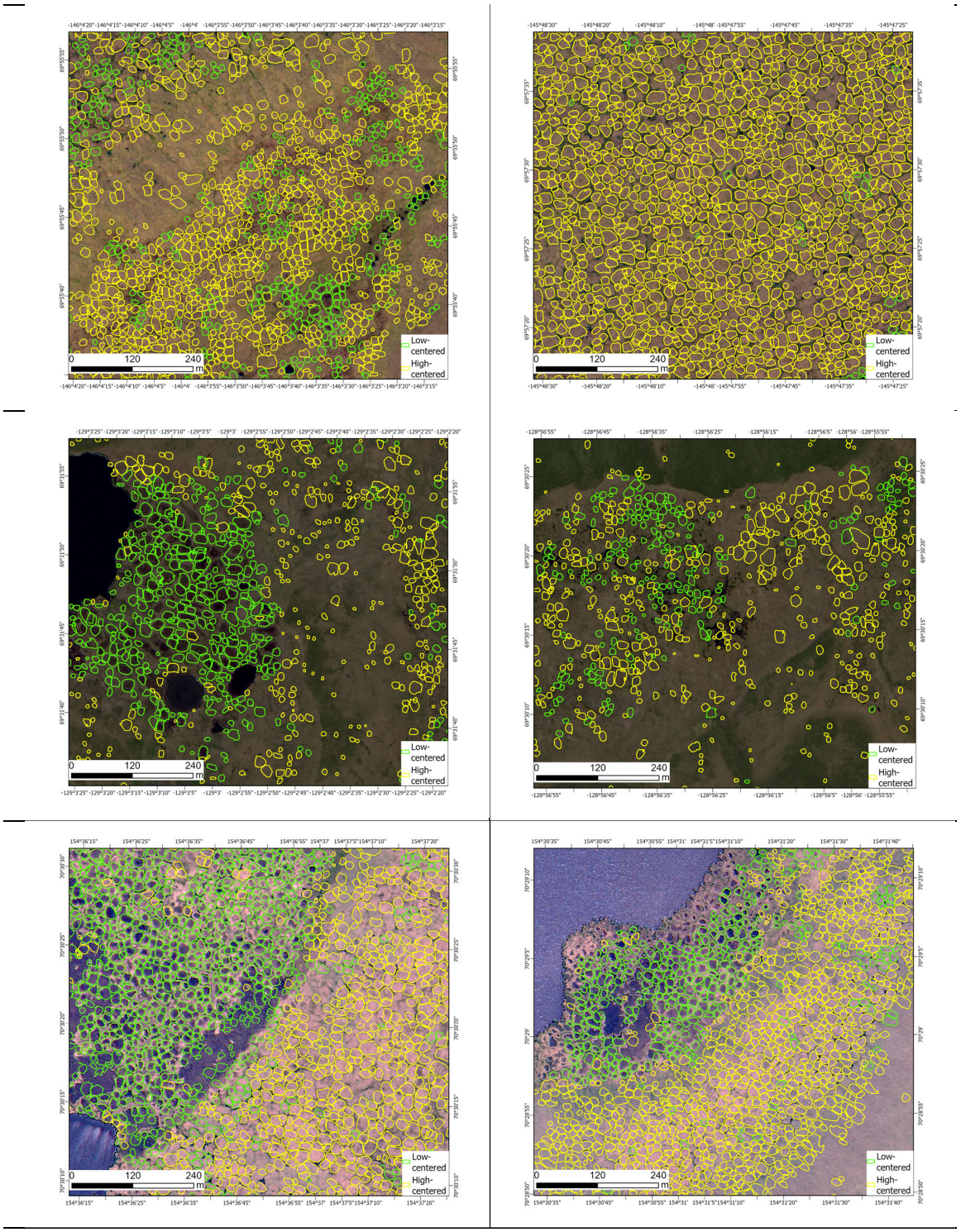


FIGURE 12. Examples of inferred IWPs superimposed on the false color composite (Satellite Images ©Maxar). Top: North Slope, Alaska located between Deadhorse and Kaktovik, approximately 25 km from the Beaufort Sea, Middle: Approximately 160 km east of Tuktoyaktuk in Northwest Territories, Canada; Bottom: Sakha Republic, Russia, approximately 50 km from the East Siberian Sea.

approach suggested in this work is to identify the available computing resources for hyperparameter tuning, rank and select the parameters based on possible impact and conduct tuning on the selected hyperparameters. Ranking parameters can be done based on prior research on similar work considering impact of parameters. The range explored for each parameter can be decided based on the available computing budget for model training considering the per-epoch resource utilization. The nested loop algorithm suggested in this work can be easily used to conduct hyperparameter

tuning without having to alter the training setup. This can be also deployed on a parallel computing environment using scripting support provided by high performance computing environments with minimal modification to the existing code base. Since the suggested approach for hyperparameter tuning is generic and independent of the DL approach used, this algorithm can be used on any CNN based or other DL algorithm for hyperparameter tuning.

C. FUTURE WORK

It is clear from the results that if DL is utilized for RS data, hyperparameter tuning should be carried out to obtain the benefits of DL for better map accuracy. Since a brute force grid search is prohibitively expensive, practitioners must investigate alternatives. A promising alternative to the mix of approaches is the use of evolutionary computing, where Darwinian evolution is used to search for the best hyperparameter combination via a population search that uses crossover and mutations to find better solutions as it evolves. Although evolutionary computing approaches have been used for hyperparameter tuning in traditional ML, the high computing cost of training for DL-based RS products has not enabled evolutionary computing in the literature, even with the availability of high-performance computing environments and applicable tools.

Tune [23], which is built on top of Ray, a distributed framework for emerging AI applications [59] is a promising approach for hyperparameter tuning where evolutionary computing is one of the options to search through the hyperparameter space. The solution also includes the ability for early stopping to truncate the DL training process if a model is not performing well based on the loss value at a given training epoch without going until the last epoch. In addition, Ray Tune provides options to specify the amount of resources to use for tuning, the hyperparameters, which range they are sampled from, and a scheduling or optimization algorithm. With this approach, a single Ray Tune job is started, and Ray deals with all scheduling and communication tasks [23]. In summary, Ray Tune can run distributed hyperparameter tuning at scale. To use a hyperparameter tuner such as Ray, a DL codebase with an appropriate interface that can communicate with the DL training module and a sufficient training budget is essential. The work presented in this paper attempts to propose a hyperparameter tuning approach that is less dependent on the training interface and also a much lower training budget compared to using Ray for tuning.

V. CONCLUSION

Although DL approaches are widely used for RS, hyperparameter tuning is an often-neglected problem dimension due to the prohibitive computing cost. The availability of large-scale distributed computing at high performance computing environments and the availability of distributing computing frameworks, provide an opportunity to explore the impact of hyperparameter tuning and improve the final product.

In this work we show the impact of using hyperparameter tuning when using DL based approaches for RS with a case example of mapping ice wedge polygons using sub-meter resolution satellite images at Pan-Arctic scale. The mAP is improved from 0.72 to 0.83 (16%) using a few minimal hyperparameter tuning options in a systematic manner, based on the availability of computing resources. In this specific case study, it could be argued that the quality of the DL capability to distinguish between high-ice and low-ice IWPs would significantly impact the proceeding scientific analysis done on the derived RS products.

Based on our RS DL earth science use case, it can be recommended to engage in hyperparameter tuning by selecting a set of high-impact parameters within the confines of the computing resources available for DL model training and tuning.

APPENDIX

A. DEGREE OF HYPERPARAMETER TUNING IN DL BASED RSI ANALYSIS

The following is a description of recent literature in RS using DL. The description is broken into 4 sections based on the degree of application of hyperparameter tuning.

1) HYPERPARAMETER TUNING: NOT MENTIONED

A comprehensive review of achievements and challenges in DL in environmental RS [26] does not discuss the issue of hyperparameter tuning as a challenge, indicating the lack of emphasis given to the impact of hyperparameter tuning. A technical review on state of the art in DL in RS [27] also does not discuss the issue of hyperparameter tuning as a challenge. A comprehensive review of DL in RS [5] with almost 200 related references does not discuss the issue of hyperparameter tuning, further indicating the lack of emphasis given to the impact of hyperparameter tuning. In this work [5], there are few references to prior works that discuss fine-tuning the DL architecture, hinting at some form of hyperparameter tuning, but the review does not give any emphasis to hyperparameter tuning. A comprehensive meta-analysis and review of DL in RS applications [28], and Challenges, Methods, Benchmarks, and Opportunities in RS with DL [29] do not mention hyperparameter tuning.

There are multiple application papers which do not indicate any hint of hyperparameter tuning or fine-tuning the systems for better results: Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data [30], Deep Learning Approach for Remote Sensing Image Classification [31], A Deep Learning Approach for Spatiotemporal Prediction of Remote Sensing Data [32], Deep Learning Based Feature Selection for Remote Sensing Scene Classification [33], Deep Learning Meets Metric Learning: Remote Sensing Image Scene Classification via Learning Discriminative CNNs [34], Deep-learning-based information mining from ocean remote-sensing imagery [35], and Geological Disaster Recognition on Optical Remote Sensing Images Using Deep Learning [36].

2) HYPERPARAMETER TUNING: MENTIONED BUT NOT DONE

Improving the Efficiency of Deep Learning Methods in Remote Sensing Data Analysis: Geosystem approach [38] describes hyperparameter tuning as an open problem and makes hyperparameters as one of the configurable options in the proposed framework, but no results are reported about using hyperparameter tuning with reported results on their application for detecting landslides from RS data.

3) HYPERPARAMETER TUNING: PARTIALLY DONE AND OR NOT REPORTED

Multimodal DL for Remote-Sensing Imagery Classification [40] mentions the importance of hyperparameters and reports using a grid search for hyperparameters but does not indicate the shape or size of the space explored, nor the accuracy variations observed from the tuning. A Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community [39] acknowledges the importance of hyperparameter tuning and reports one prior work [42] where a limited number of hyperparameters are explored and tuned. The importance of hyperparameter optimization is indicated in [41], and the work includes a discussion on using multiple configurations of hyperparameters to arrive at the final solution. However, it does not report any results on hyperparameter optimization trials.

4) HYPERPARAMETER TUNING: DONE AND REPORTED

In [43], the authors apply DL for landcover classification and explore the various options available for hyperparameter optimization, including grid search, random search, Bayesian optimization, hyperband, and a hybrid approach [22] that combines Bayesian with hyperband and also considers the cost of hyperparameter tuning in the literature review. In [43], the authors also introduce Ray Tune [23] as a possible platform that can be utilized to do hyperparameter optimization and exploit parallel computing if available. The work in [43] describes a methodological approach for hyperparameter tuning, where the authors have restricted the search to an intuitively selected subset of parameters, while also reporting the results, helping the readers to understand the selection process and the impact of various parameters on the outcome.

In [45], the authors acknowledge the importance of hyperparameter tuning and report results obtained from a grid search. It is important to note that the non-DL baseline models (LR, SVM, RF) used to compare are also used after executing hyperparameter tuning, demonstrating a deeper understanding of the impact of hyperparameter tuning on ML. But in [45], only the best-performing configurations are reported for the DL hyperparameter tuning done for the network layers, initial filters, L2 Regularization, and Learning Rate. No results are reported on the impact of hyperparameter tuning, but a discussion is included regarding some of the parameters that are explored and the range and step size of the values explored when tuning [45].

In [46], authors acknowledge the importance of hyperparameter tuning and report the optimum results obtained. The selected parameters are mentioned, but the method employed to execute the search is not indicated. Readers may have to assume it was a grid search, considering the limited number of parameters (network, iterations, batch size, patch size, and learning rate) that are explored in this work.

In [47], the authors introduce changing a hyperparameter, specifically the batch size, to improve the training time for DL-based RS applications. In [47], the authors emphasize the use of hyperparameter tuning in DL and also employ Ray Tune [23] with a restricted set of parameters to explore the hyperparameter space.

In [44], a comparison is done on a single family of CNN-based DL models for RS data with the hyperparameters locked, as the authors are not interested in finding the best solution but in comparing models. The work [44] acknowledges the importance of hyperparameter tuning and comparing a single parameter (model) while keeping the rest of the parameters locked to gain an insight on how the respective parameter impacts the outcome.

ACKNOWLEDGMENT

The authors would like to thank the Polar Geospatial Center, University of Minnesota, for imagery support and also would like to thank the Texas Advanced Computing Center for high performance computing support.

REFERENCES

[1] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for UAVs: Current developments and trends," *J. Intell. Robotic Syst.*, vol. 87, no. 1, pp. 141–168, Jul. 2017.

[2] A. K. Liljedahl, "Permafrost discovery gateway: A web platform to enable discovery and knowledge-generation of permafrost big imagery products," in *Proc. EPIC3AGU Fall Meeting*, San Francisco, CA, USA, Dec. 2019, pp. 1–12. Accessed: May 10, 2023. [Online]. Available: https://epic.awi.de/id/eprint/50806/

[3] A. Bykova. *Permafrost Thaw in a Warming World: The Arctic Institute's Permafrost Series Fall-Winter 2020*. The Arctic Institute—Center for Circumpolar Security Studies. Accessed: Apr. 28, 2023. [Online]. Available: https://www.thearcticinstitute.org/permafrost-thaw-warming-world-arctic-institute-permafrost-series-fall-winter-2020/

[4] M. Udawalpola, A. Hasan, A. K. Liljedahl, A. Soliman, and C. Witharana, "Operational-scale geoai for pan-arctic permafrost feature detection from high-resolution satellite imagery," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. XLIV-M-3-2021, pp. 175–180, Aug. 2021, doi: 10.5194/isprs-archives-xliv-m-3-2021-175-2021.

[5] X. X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017, doi: 10.1109/MGRS.2017.2762307.

[6] N. R. P. Salinas, M. Baratchi, J. N. van Rijn, and A. Vollrath, "Automated machine learning for satellite data: Integrating remote sensing pre-trained models into AutoML systems," in *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track*, Y. Dong, N. Kourtellis, B. Hammer, and J. A. Lozano, Eds. Cham, Switzerland: Springer, 2021, pp. 447–462, doi: 10.1007/978-3-030-86517-7_28.

[7] K. Wang, S. E. Franklin, X. Guo, Y. He, and G. J. McDermid, "Problems in remote sensing of landscapes and habitats," *Prog. Phys. Geography, Earth Environ.*, vol. 33, no. 6, pp. 747–768, Dec. 2009, doi: 10.1177/0309133309350121.

[8] T. Blaschke, G. J. Hay, M. Kelly, S. Lang, P. Hofmann, E. Addink, R. Q. Feitosa, F. van der Meer, H. van der Werff, F. van Coillie, and D. Tiede, "Geographic object-based image analysis—Towards a new paradigm," *ISPRS J. Photogramm. Remote Sens.*, vol. 87, pp. 180–191, Jan. 2014, doi: 10.1016/j.isprsjprs.2013.09.014.

[9] G. J. Hay, T. Blaschke, D. J. Marceau, and A. Bouchard, "A comparison of three image-object methods for the multiscale analysis of landscape structure," *ISPRS J. Photogramm. Remote Sens.*, vol. 57, nos. 5–6, pp. 327–345, Apr. 2003, doi: 10.1016/s0924-2716(02)00162-4.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," 2012, *arXiv:1206.5538*.

[11] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[12] G. Koppe, "Deep learning for small and big data in psychiatry," *Neuropsychopharmacology*, vol. 46, no. 1, pp. 90–176, 2021.

[13] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, Sep. 1999, pp. 1150–1157.

[14] L. Wu, S. C. H. Hoi, and N. Yu, "Semantics-preserving bag-of-words models and applications," *IEEE Trans. Image Process.*, vol. 19, no. 7, pp. 1908–1920, Jul. 2010.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[17] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2018, pp. 270–279.

[18] Q. Xie, M. Suvarna, J. Li, X. Zhu, J. Cai, and X. Wang, "Online prediction of mechanical properties of hot rolled steel plate using machine learning," *Mater. Des.*, vol. 197, Jan. 2021, Art. no. 109201, doi: 10.1016/j.matdes.2020.109201.

[19] S. Amari, "A theory of adaptive pattern classifiers," *IEEE Trans. Electron. Comput.*, vol. EC-16, no. 3, pp. 299–307, Jun. 1967, doi: 10.1109/PGEC.1967.264666.

[20] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," Jan. 2019, *arXiv:1709.06560*.

[21] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," 2017, *arXiv:1707.05589*.

[22] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," 2018, *arXiv:1807.01774*.

[23] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," 2018, *arXiv:1807.05118*.

[24] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106622, doi: 10.1016/j.knosys.2020.106622.

[25] N. Audebert, B. Le Saux, and S. Lefevre, "Deep learning for classification of hyperspectral data: A comparative review," *IEEE Geosci. Remote Sens. Mag.*, vol. 7, no. 2, pp. 159–173, Jun. 2019, doi: 10.1109/MGRS.2019.2912563.

[26] Q. Yuan, H. Shen, T. Li, Z. Li, S. Li, Y. Jiang, H. Xu, W. Tan, Q. Yang, J. Wang, J. Gao, and L. Zhang, "Deep learning in environmental remote sensing: Achievements and challenges," *Remote Sens. Environ.*, vol. 241, May 2020, Art. no. 111716, doi: 10.1016/j.rse.2020.111716.

[27] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, Jun. 2016, doi: 10.1109/MGRS.2016.2540798.

[28] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS J. Photogramm. Remote Sens.*, vol. 152, pp. 166–177, Jun. 2019, doi: 10.1016/j.isprsjprs.2019.04.015.

[29] G. Cheng, X. Xie, J. Han, L. Guo, and G.-S. Xia, "Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 3735–3756, 2020, doi: 10.1109/JSTARS.2020.3005403.

[30] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, "Deep learning classification of land cover and crop types using remote sensing data," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 5, pp. 778–782, May 2017, doi: 10.1109/LGRS.2017.2681128.

[31] A. Ben Hamida, A. Benoit, P. Lambert, and C. Ben Amar, "3-D deep learning approach for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4420–4434, Aug. 2018, doi: 10.1109/TGRS.2018.2818945.

[32] M. Das and S. K. Ghosh, "Deep-STEP: A deep learning approach for spatiotemporal prediction of remote sensing data," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 1984–1988, Dec. 2016, doi: 10.1109/LGRS.2016.2619984.

[33] Q. Zou, L. Ni, T. Zhang, and Q. Wang, "Deep learning based feature selection for remote sensing scene classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 11, pp. 2321–2325, Nov. 2015, doi: 10.1109/LGRS.2015.2475299.

[34] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 5, pp. 2811–2821, May 2018, doi: 10.1109/TGRS.2017.2783902.

[35] X. Li, B. Liu, G. Zheng, Y. Ren, S. Zhang, Y. Liu, L. Gao, Y. Liu, B. Zhang, and F. Wang, "Deep-learning-based information mining from ocean remote-sensing imagery," *Nat. Sci. Rev.*, vol. 7, no. 10, pp. 1584–1605, Oct. 2020, doi: 10.1093/nsr/nwaa047.

[36] Y. Liu and L. Wu, "Geological disaster recognition on optical remote sensing images using deep learning," *Proc. Comput. Sci.*, vol. 91, pp. 566–575, Jan. 2016, doi: 10.1016/j.procs.2016.07.144.

[37] G. J. Reynolds, C. E. Windels, I. V. MacRae, and S. Laguette, "Remote sensing for assessing rhizoctonia crown and root rot severity in sugar beet," *Plant Disease*, vol. 96, no. 4, pp. 497–505, Apr. 2012.

[38] S. A. Yamashkin, A. A. Yamashkin, V. V. Zanozin, M. M. Radovanovic, and A. N. Barmin, "Improving the efficiency of deep learning methods in remote sensing data analysis: Geosystem approach," *IEEE Access*, vol. 8, pp. 179516–179529, 2020, doi: 10.1109/ACCESS.2020.3028030.

[39] J. E. Ball, D. T. Anderson, and C. S. Chan, "Comprehensive survey of deep learning in remote sensing: Theories, tools, and challenges for the community," *J. Appl. Remote Sens.*, vol. 11, no. 4, Sep. 2017, Art. no. 042609, doi: 10.1117/1.jrs.11.042609.

[40] D. Hong, L. Gao, N. Yokoya, J. Yao, J. Chanussot, Q. Du, and B. Zhang, "More diverse means better: Multimodal deep learning meets remote-sensing imagery classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 5, pp. 4340–4354, May 2021, doi: 10.1109/TGRS.2020.3016820.

[41] F. P. S. Luus, B. P. Salmon, F. van den Bergh, and B. T. J. Maharaj, "Multiview deep learning for land-use classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2448–2452, Dec. 2015, doi: 10.1109/LGRS.2015.2483680.

[42] Y. Zhong, F. Fei, Y. Liu, B. Zhao, H. Jiao, and L. Zhang, "SatCNN: Satellite image dataset classification using agile convolutional neural networks," *Remote Sens. Lett.*, vol. 8, no. 2, pp. 136–145, Feb. 2017, doi: 10.1080/2150704x.2016.1235299.

[43] V. Yaloveha, A. Podorozhniak, and H. Kuchuk, "Convolutional neural network hyperparameter optimization applied to land cover classification," *RADIOELECTRONIC Comput. Syst.*, vol. 1, pp. 115–128, Feb. 2022, doi: 10.32620/reks.2022.1.09.

[44] F. I. Diakogiannis, F. Waldner, P. Caccetta, and C. Wu, "ResUNet–A: A deep learning framework for semantic segmentation of remotely sensed data," *ISPRS J. Photogramm. Remote Sens.*, vol. 162, pp. 94–114, Apr. 2020, doi: 10.1016/j.isprsjprs.2020.01.013.

[45] S. Wang, W. Chen, S. M. Xie, G. Azzari, and D. B. Lobell, "Weakly supervised deep learning for segmentation of remote sensing imagery," *Remote Sens.*, vol. 12, no. 2, p. 207, Jan. 2020, doi: 10.3390/rs12020207.

[46] S.-H. Lee, K.-J. Han, K. Lee, K.-J. Lee, K.-Y. Oh, and M.-J. Lee, "Classification of landscape affected by deforestation using high-resolution remote sensing data and deep-learning techniques," *Remote Sens.*, vol. 12, no. 20, p. 3372, Oct. 2020, doi: 10.3390/rs12203372.

[47] M. Aach, R. Sedona, A. Lintermann, G. Cavallaro, H. Neukirchen, and M. Riedel, "Accelerating hyperparameter tuning of a deep learning model for remote sensing image classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2022, pp. 263–266, doi: 10.1109/IGARSS46834.2022.9883257.

[48] C. Witharana, M. R. Udawalpola, A. K. Liljedahl, M. K. W. Jones, B. M. Jones, A. Hasan, D. Joshi, and E. Manos, "Automated detection of retrogressive thaw slumps in the high Arctic using high-resolution satellite imagery," *Remote Sens.*, vol. 14, no. 17, p. 4132, Aug. 2022, doi: 10.3390/rs14174132.

[49] A. E. Steedman, T. C. Lantz, and S. V. Kokelj, "Spatio-temporal variation in high-centre polygons and ice-wedge melt ponds, Tuktoyaktuk coastlands, Northwest territories," *Permafrost Periglacial Processes*, vol. 28, no. 1, pp. 66–78, Jan. 2017.

[50] A. K. Liljedahl, J. Boike, R. P. Daanen, A. N. Fedorov, G. V. Frost, G. Grosse, L. D. Hinzman, Y. Iijma, J. C. Jorgenson, N. Matveyeva, M. Necsoiu, M. K. Raynolds, V. E. Romanovsky, J. Schulla, K. D. Tape, D. A. Walker, C. J. Wilson, H. Yabuki, and D. Zona, "Pan-arctic ice-wedge degradation in warming permafrost and its influence on Tundra hydrology," *Nature Geosci.*, vol. 9, no. 4, pp. 312–318, Apr. 2016.

[51] M. Kanevskiy, Y. Shur, T. Jorgenson, D. R. N. Brown, N. Moskalenko, J. Brown, D. A. Walker, M. K. Raynolds, and M. Buchhorn, "Degradation and stabilization of ice wedges: Implications for assessing risk of thermokarst in northern Alaska," *Geomorphology*, vol. 297, pp. 20–42, Nov. 2017.

[52] C. J. Burke, P. D. Aleo, Y.-C. Chen, X. Liu, J. R. Peterson, G. H. Sembroski, and J. Y.-Y. Lin, "Deblending and classifying astronomical sources with mask R-CNN deep learning," *Monthly Notices Roy. Astronomical Soc.*, vol. 490, no. 3, pp. 3952–3965, Dec. 2019.

[53] *Mask_RCNN/Samples/Balloon at Master.Matterport/Mask_RCNN*. GitHub. Accessed: Jun. 8, 2023. [Online]. Available: https://github.com/matterport/Mask_RCNN

[54] D. Stanzione, J. West, R. T. Evans, T. Minyard, O. Ghattas, and D. K. Panda, "Frontera: The evolution of leadership computing at the national science foundation," in *Proc. Pract. Exper. Adv. Res. Comput. (PEARC)*. New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 106–111, doi: 10.1145/3311790.3396656.

[55] Y. Wu, L. Liu, J. Bae, K.-H. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, and Q. Zhang, "Demystifying learning rate policies for high accuracy training of deep neural networks," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 1971–1980, doi: 10.1109/BigData47090.2019.9006104.

[56] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Cham, Switzerland: Springer, 2017, pp. 240–248, doi: 10.1007/978-3-319-67558-9_28.

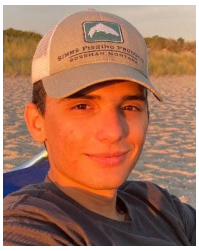
[57] K. Nakamura, B. Derbel, K.-J. Won, and B.-W. Hong, "Learning-rate annealing methods for deep neural networks," *Electronics*, vol. 10, no. 16, p. 2029, Aug. 2021, doi: 10.3390/electronics10162029.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[59] P. Moritz, R. Nishihara, S. Wang, and A. Tumanov, "Ray: A distributed framework for emerging AI applications," in *Proc. 13th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2018, pp. 561–577.

AMAL S. PERERA (Member, IEEE) received the Ph.D. degree in computer science. He is currently a Senior Lecturer in computer science with the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka. He is also a Data Scientist with the University of Connecticut as an invited Postdoctoral Research Scholar. He is also working on developing and deploying deep learning-based computer vision models on big data on high performance computing environments. He has multiple research publications and a co-owner of a U.S. patent. His research interests include data science, data mining, database systems, and software engineering. He won the ACM KDD Cup for Data Mining, in 2006.

CHANDI WITHARANA is a Professor at the University of Connecticut, with a Ph.D. in remote sensing. His research interests include capture the methodological developments and adaptations to unseal faster, deeper, and more accurate analysis of large volumes of high-resolution remote sensing data. He conducts interdisciplinary remote sensing research with high international visibility, speaking equally to the transformational uses of remote sensing in environmental, industrial, and agricultural applications.



ELIAS MANOS is currently pursuing the M.S. degree in remote sensing with the Department of Natural Resources, University of Connecticut. He is also a Geospatial Deep Learning Practitioner with a background in GIS/remote sensing. He is a Team Member of the Permafrost Discovery Gateway Project (funded by the NSF's Navigating the New Arctic Program). His research interests include advancing remote sensing of Arctic permafrost regions by developing deep learning approaches to map the natural and built environments from high-resolution satellite imagery with unprecedented spatial and thematic detail. He is also developing a deep learning approach to map Arctic infrastructure at a <1 m spatial resolution and circumpolar scale from Maxar satellite imagery to enable accurate assessment of infrastructure damage risk in the face of climate change-induced permafrost degradation.

ANNA K. LILJEDAHL is currently an Associate Scientist with the Woodwell Climate Research Center. She is also the Project Lead of the Permafrost Discovery Gateway Team. Her work is rooted in a life-long love of water and a deep sense of connection to Alaska's environment and communities. Her research interests include how climate change is altering the storage and movement of water in Arctic ecosystems. She has probed the impact of glacial melt and permafrost thawing on natural processes, like ponding and runoff and built infrastructure, including hydropower. Her work combines field measurements with remote sensing data to produce high-resolution computer models at watershed scales. Through projects like the Permafrost Discovery Gateway, she strives to expand access to information and to expedite knowledge-generation from big data to serve earth scientists and communities facing climate impacts.