Constructing a Large-scale Landslide Database Across Heterogeneous Environments Using Task-Specific Model Updates

Savinay Nagendra[1], Daniel Kifer[1], Benjamin Mirus[3], Te Pei[2], Kathryn Lawson[2], Srikanth Banagere Manjunatha[1], Weixin Li[2], Hien Nguyen[4], Tong Qiu[2], Sarah Tran[5], and Chaopeng Shen[2]

[1]Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA
[2]Civil and Environmental Engineering, Pennsylvania State University, University Park, PA, USA
[3*]U.S. Geological Survey, Landslide Hazards Program, Golden, CO, USA
[4]School of Science, Engineering, and Technology, Pennsylvania State University Harrisburg, Harrisburg, PA, USA
[5]Google Inc., Mountain View, CA, USA

***Abstract*—Preparation and mitigation efforts for widespread landslide hazards can be aided by a large-scale, well-labeled landslide inventory with high location accuracy. Recent small-scale studies for pixel-wise labeling of potential landslide areas in remotely-sensed images using deep learning (DL) showed potential but were based on data from very small, homogeneous regions with unproven model transferability. In this paper we consider a more realistic and practical setting for large-scale heterogeneous landslide data collection and DL-based labeling. In this setting, remotely sensed images are collected sequentially in temporal batches, where each batch focuses on images from a particular ecoregion, but different batches can focus on different ecoregions with distinct landscape characteristics. For such a scenario, we study the following questions: (1) How well do DL models trained in homogeneous regions perform when they are transferred to different ecoregions, (2) Does increasing the spatial coverage in the data improve model performance in a given ecoregion (even when the extra data do not come from the ecoregion), and (3) Can a landslide pixel labeling model be incrementally updated with new data, but without access to the old data and without losing performance on the old data (so that researchers can share models obtained from proprietary datasets)? We address these questions by extending the *Learning without Forgetting framework*, which is used for incremental training of image classification models, to the setting of incremental training of semantic segmentation models (e.g., identifying all landslide pixels in an image). We call the resulting extension Task-Specific Model Updates (TSMU). The goal of this framework is to continually update a (landslide) semantic segmentation model with data from new ecoregions without having to revisit data from old ecoregions and without losing performance on them.**

Using the TSMU framework, we conduct extensive experiments on four ecoregions in the United States to address the aforementioned questions. The results, ordered from unexpected to most unexpected, show that (i) a DL model trained on data from one ecoregion can work well on that ecoregion, but (ii) it is not expected to perform well on ecoregions it has not seen before, but (iii) data from other ecoregions can help improve the model's performance on the original ecoregion. In other words, if one has an ecoregion of interest, one could still collect data both inside and outside that region to improve model performance on the ecoregion of interest. Furthermore, if one has many ecoregions of interest, data from all of them are needed.

Another interesting feature of the TSMU framework is that although it never revisits data from old ecoregions when it is being trained on new ecoregions, its final performance nearly matches the performance of the computationally expensive ideal setting: simultaneous training on all of the data. This property will allow us to make our model public, so that other researchers can update it with their ecoregions, without losing performance on older data. With these features, the TSMU framework can be used to aid in the creation of new landslide inventories or expanding existing datasets, and also to rapidly develop hazard maps for situational awareness following a widespread landsliding event.

Our dataset and code is made publicly accessible upon acceptance here - https://github.com/deepLDB/landslide-detection

***Index Terms*—Landslides, semantic segmentation, deep learning, ecoregions, continual learning, domain adaptation, catastrophic forgetting.**

I. Introduction

LANDSLIDES are one of the most widespread geologic hazards, with over 300M people exposed and over 66M people living in high-risk areas [1]. Globally, landslides cause thousands of deaths each year (≈4,164 in 2017 alone [2]), displacement of communities, and destruction of infrastructure and habitable lands. Moreover, climate change is projected to induce more frequent extreme rainfalls and wildfires, which are expected to result in more landslides and landslide-related casualties [3], [4].

Long-term efforts to mitigate the effects of landslides require evaluating landslide susceptibility [5]. Planning and prediction capabilities can be greatly enhanced by large-scale databases of landslide events with accurate location information [6]. For example, in the United States, the U.S. Geological Survey (USGS) maintains a database of landslide reports with approximate locations and times, but no images (https://doi.org/10.5066/P9E2A37P). The database can be found on our GitHub link for data. It is the most extensive dataset of its kind, but it relies on intensive manual effort to map landslides through field investigations or remote sensing and is still very much incomplete. Furthermore, its spatial location accuracy leaves much to be desired for planning

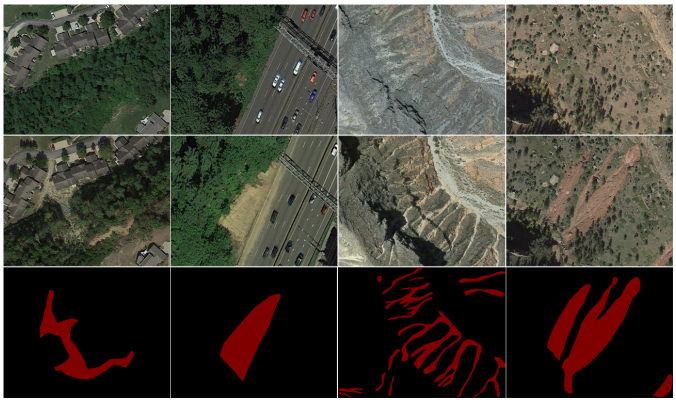

Fig. 1: **Sample bi-temporal pairs of images and corresponding human annotations from four ecoregions.** Pre-event (top row), post-event (middle row) images and human annotated labels (bottom row) from four ecoregions of the United States in our dataset (left to right): Eastern Temperate Forests (ETF), Marine West Coast Forests (MWCF), North American Deserts (NAD) and Northwestern Forested Mountains (NWFM). Landslide areas in images may be confused with human activities such as logging or construction, which makes it hard for human annotators to discern while labeling. The excel file containing the coordinates of the landslides used in this figure will be available in our GitHub link for data.

purposes: recorded coordinates are often not precise enough for overlaying other datasets. While some entry can be verified manually from high-resolution satellite images, this process requires substantial effort and prevents scaling up. These issues present both an opportunity and a challenge for scalability and coverage, as recently summarized by the creators of the USGS database: "our current ability to understand landslide hazards at the national scale is limited, in part because spatial data on landslide occurrence across the U.S. varies greatly in quality, accessibility, and extent" [7].

A grand challenge to a uniform and semi-automated approach to map landslides globally is the wide ranging modes of slope failure, from shallow and rapid failures of soil material, to rock falls, and gradual deep-seated slope movements or deformation. Rock slope and deep-seated failures are difficult to identify with imagery based mapping because changes in vegetation and surface properties are often not detectable. Instead, we focus on detection of the rainfall and earthquake triggered shallow landslides or debris flows (i.e., space-visible landslides) that can be readily detected from satellite imagery via the resulting disturbance to vegetation. Additionally, due to their sudden onset and typically rapid movement, these landslides are the most damaging and deadly.

Given the increasing availability of high-resolution satellite imagery, a path is now open to collecting information about space-visible landslide events by having machine learning algorithms scan the images. Several recent efforts [8]–[11] have considered the use of deep learning for landslide segmentation (also known as landslide mapping) – identifying the pixels in a satellite image that correspond to landslides. However, due to the difficulty of labeling training data, all of these studies only focused on a single small region and had limited testing data (for instance, the study in [9] only had three testing images). As we will show later, this practice may reduce the robustness of the model and also undermine the potential accuracy of the model even for the region of interest.

Space-visible landslides are often irregular in shape, are not always easy to discern in an image, and may be confused with human activities such as logging or construction (Figure 1). Thus, even human annotators need explicit training. It takes trained annotators several minutes to locate a landslide in an image, and then several more minutes to process the image and map the landslide polygon. Even trained annotators may not be able to fully discern between erosional features and landslides, and confirming this distinction would require high-resolution topographic data, stereographic imagery, or better yet in-person field characterization. Such data are not as widely available as the orthorectified satellite images used here. For these reasons, building a global image database for space-visible landslides by hand is infeasible. The training required for human labelers also rules out crowd-sourcing platforms like Amazon Mechanical Turk [12]. Given the necessary resources and remaining uncertainty with any inventory of space-visible landslides, automated and semi-automated approaches for constructing landslide databases appear more promising [13].

We consider a practical large-scale data collection scenario where remotely-sensed images are collected over time in batches. Within a batch, images usually come from the same ecoregion (an area where ecosystems, soil, and the environment are generally similar). Different batches may come from different ecoregions with possibly drastically different environmental features. Space-visible landslides are often identifiable due to the disturbance of the land surface and vegetation, and these features vary in appearances among different ecoregions. In this paper, we use data from four ecoregions - Eastern Temperate Forests (ETF), Marine West Coast Forests (MWCF), North American Deserts (NAD), and Northwestern Forested Mountains (NWFM). As can be seen from Figure 1, landslides in different ecoregions differ in color, texture, landslide size, and visual contrast with the background. Each ecoregion has its own distinct data distribution due to visually dissimilar landscape characteristics (see data section) – this is often referred to as a domain shift in the machine learning literature [14], [15].

In such a scenario, the first question we ask is "**does a model trained on a small homogeneous ecoregion** [8], [9] **generalize to other ecoregions for which it has not been trained?**" To test this, we perform spatial Leave-One-Out (LOO) experiments where data from multiple ecoregions were used for training and data points from another ecoregion are withheld for testing. The selection was rotated so that each ecoregion had the chance to serve as a test region. We examined how model accuracy responded when models were tested on an ecoregion that is characteristically different from the training set.

The second question is "**does increasing the spatial coverage and diversity of data improve model performance in a given ecoregion?**" For each ecoregion R, we compared models trained on only region R to those trained on region

R plus data from other ecoregions to increase diversity. We evaluate how well they perform specifically on that ecoregion R. The answer to this question leads to an informed strategy regarding data collection, i.e., if a diverse training dataset significantly improves the results compared to local data alone, then expanding the spatial extent of our data collection beyond the specific region of interest would be beneficial.

The third question is **"what is an efficient and highly-performant way to incorporate new batches of geospatial training data from different environments to update a semantic segmentation model?"** We propose *Task-Specific Model Updates* (TSMU), a continual learning [16] semantic segmentation strategy, which is an extension of *Learning without Forgetting* (LwF) [17], which was used for image classification. TSMU has the following features:

1) It uses only the data from the new batch of data (without revisiting old data) for training for computational efficiency. This feature allows models to be shared between research groups without having to share training data that may be proprietary.
2) Updates of the model parameters will not cause *catastrophic forgetting* [18] (a common machine learning phenomenon where performance on old data degrades). In fact, in our experiments we observed that accuracy over old ecoregions improves as new ecoregions are added.
3) Addresses the domain shift [14] problem in semantic segmentation by modeling features that are common to multiple ecoregions and also separately modeling ecoregion-specific idiosyncracies.

The proposed method drastically reduces the computational time of efficiently incorporating sequentially acquired batches of training data while nearly matching the performance of a computationally expensive "ideal" model that jointly process all of the data.

II. Related Work

In this section, we describe competing methods in transfer learning and continual learning used for semantic segmentation.

A. Landslide Mapping

Detection and classification of landslides is critical for hazard analysis [19]–[21]. Landslides and associated erosion can be identified by detecting the disturbance of earth surfaces. Aerial photographs and satellite images have been widely used for landslide detection [22]. We briefly review some commonly used approaches for landslide mapping from remote sensing images.

1) Classical Approach: Landslides are rare spatially and are difficult to spot from large swaths of satellite images [22], [23]. Conventional computer vision techniques [22] are easily affected by noise [24] and require human intervention for each scene. Two conventional approaches are commonly used in the literature [25]: (1) pixel-based methods, which use spectral information to detect pixels corresponding to landslides from images (e.g., [26]–[29]); and (2) object-based image analysis, which uses both spectral and spatial information (e.g., [23], [30]). These two approaches are commonly integrated with image change detection to identify landslides from multi-temporal remote sensing images [31]. Machine learning (ML) techniques such as logistic regression, support vector machines, and random forests have also been used to improve model performance for both approaches [26], [32], [33]. However, both approaches require extensive parameter tuning, feature extraction, and post-processing, limiting their application at large scales.

2) Deep Learning based Approach: Deep learning (DL) has been already used for landslide mapping [11], [25], [34]–[40]. Deep learning techniques do not require hand-crafted features as they perform automatic feature engineering directly from satellite imagery. Hu *et al.* [35] used landslide images from multiple seasons from LandSat satellite and digital elevation model (DEM) data to extract landslides using three ML models. Prakash *et al.* [11] used Sentinel-2 images, DEM data from light detection and ranging (Lidar), and a derived normalized difference vegetation index (NDVI) layer to map landslides using a modified U-Net model [41], [42]. Due to the sparse availability of high spatial resolution DEMs, especially in mountainous regions, many times only spectral bands of satellite images were used. Qi *et al.* [25] proposed a deep network called ResU-Net, which uses residual blocks [43] in the encoder of U-Net architecture. The authors claimed that the residual block can alleviate the data sparsity problems [25]. Soares et al. [44] used U-Net to automatically segment landslides in the city of Nova Friburgo. Lei *et al.* [45] compared three DL models – Fully Convolutional Network, Fully-connected Deep Neural Network, and U-Net – to detect landslide areas, concluding that U-Net had the best performance. Prakash *et al.* [11] used a modified U-Net model with ResNet34 blocks for feature extraction for semantic segmentation of landslides at a regional scale using Earth observation data. With extensive usage and success of the U-Net architecture with residual building blocks for semantic segmentation of aerial imagery, we build our network with the same backbone.

B. Limitations of Homogeneous training data

While the above mentioned studies showed early success, each of them focused on a small, homogeneous region. For example, Bragagnolo *et al.* [10] used a standard U-Net architecture on Landsat images from Nepal with 10 testing images. Amatya *et al.* [46] and Qi *et al.* [25] respectively worked on a small region in Nepal and Tianshui city in the Gansu province of China. These small scales might have been inherently limited by the scopes of these studies, but regardless, two issues arise: (i) it is uncertain whether the models trained in these regions are applicable outside of the training region; (ii) it is unclear if these locally trained models achieve optimal performance when they are specialized in a small, homogeneous region [47]. For building a global landslide database with semi-automated DL base labeling, it is necessary for the model to be robust to changes in data distributions. Oftentimes, ML model performance has improved

when exposed to slightly different distributions which, by virtue of contrast, better inform the model of the nature of the problem [48]. To the best of our knowledge, no study thus far has examined the effects of including images from different regions on vision model performance for landslide mapping.

C. Sequential Data Collection & Continual Learning Approaches

As discussed earlier, in a realistic setting, training data arrives sequentially in temporal batches, which may undergo a domain shift [49]. We use the term *task* to define landslide mapping in one ecoregion. A DL model trained on one task often suffers from catastrophic forgetting [18] when subsequently trained on a new task, performing poorly on the old task. This happens when continuously acquiring incrementally available information from non-stationary data distributions, and the parameters in the model change to meet the objectives of the new task. Hence, there is a need for continually updating the model while preserving the old information (also known as *learned representations*). Taufique *et al.* [50] proposed a continual [51]–[53] domain adaptation technique where data distillation [54] with selective replay mechanism is used to transfer learned representations across shared layers. Their technique uses samples of data from old tasks.

The problem of catastrophic forgetting [18] and the continual learning paradigm have been extensively studied. Continual Learning [55], [56] approaches can be classified into the following categories as described below.

1) Replay-based methods: These methods span from naive rehearsal [52] algorithms, which store old data to pseudo-rehearsal methods where generative models are used to approximate previous data samples [57].

2) Regularization-based methods: Data-focused [17], [58] regularization approaches use data distillation to utilize the knowledge of old tasks to enhance the performance of the model. Prior-focused [18], [59], [60] methods use regularization loss functions that penalize the shift of important parameters in the network. Importance weights are usually computed using an unsupervised approach, where the sensitivity of the network's output to change in its parameters is measured [60].

3) Parameter Isolation-based methods: These methods [18], [59]–[62] freeze the parameters that are specific to old tasks. This is achieved by dynamic extension of the network [17], [63] or by a fixed architecture [64]. These methods focus on calculation of parameter importance metrics. The path integral [59] method calculates the importance of each parameter for a particular task based on a loss function. Incremental moment matching [61] proposes a merge of layers after learning new tasks. Elastic weight consolidation [18] performs sequential Bayesian estimation and uses second order derivative of parameters as a metric of importance. This method, however, assumes that the parameters for each task are independent, which limits the performance when there is a significant domain shift in temporal flow of data.

4) Network expansion-based methods: These methods [65]–[71] assign parameter subsets to different tasks. Expert Gate [66] trains an expert network for each task and learns an autoencoder gate to activate corresponding network parameters according to the input task. Piggyback [71] learns binary masks to create subnets for each task. These networks are useful when dealing with continual learning problems where the data flow undergoes substantial domain shifts. These methods also do not assume independence between shared parameters, which makes it more flexible to design.

Learning without forgetting (LwF) [17] is a multi-task learning training strategy that is a combination of data-focused regularization and dynamic parameter isolation. Proposed for classification problems, LwF adds a set of new outputs for each new task by increasing the size of the final layer of the network (and thus each task is associated with a set of output nodes). When a datum for a new task arrives, the LwF mechanism runs it through the network and stores the resulting output for all prior task-specific nodes. It then modifies the network parameters so that the new-task output nodes improve accuracy on the task, while trying to prevent the output of the prior task nodes from changing. This is done by using knowledge distillation loss [54] in the training objective function. Image classification (e.g., does an image contain a landslide or not) is a different and easier problem than image segmentation (labeling which pixels belong to landslides) and requires different DL architectures. We thus extend LwF to image segmentation and the refer to this extension as TSMU. We compare it with several baselines and other state-of-the-art methods:

1) Feature Extraction: This is a method [72], [73] where outputs of the lower levels of a trained network (such as the convolutional layers) are considered to be feature generators for an input image (because convolutional layers are thought to extract important visual information from images). When data for a new ecoregion arrive, these features layers are re-used as the bottom half of a new (ecoregion-specific) network and only parameters in the top half are trained. Feature extraction can be further improved by subsequent *fine-tuning*, which we describe next.
2) Fine Tuning: If a network is trained for a specific task and new task-specific parameters are added, fine-tuning [74] continues training the whole (or specific layers of the) network over the new data with a small learning rate. This allows the network to leverage its existing knowledge when learning about the new task. The literature shows that fine-tuning often performs better on the new task data than feature extraction [74], [75] and has better performance than retraining the task-specific network from randomly initialized weights [76], [77]. The low learning rate is crucial and is used as an attempt to preserve the knowledge learned in the original tasks.
3) Joint Training: In this method [78], data from all of the tasks are available all at once when training the network. In this case, the network typically has a set of shared parameters as well as task-specific parameters that are optimized together. This computationally expensive option is considered "ideal" in terms of expected accuracy, and is often the default approach for DL practitioners

when all data are available at once. Thus, the goal of methods designed for temporal batches of data is to try to match the performance of joint training, despite having to operate in a more restricted setting than joint training.

4) Other competing methods: Knowledge distillation binary cross entropy (KD-BCE) [79] is a regularization-based continual learning strategy that performs multi-class remote sensing semantic segmentation. They use a replay based mechanism to sample data from old classes. However, the method assumes a class continual learning setting where the new class and the old class data come from the same underlying data distribution. Feng *et al.* [80] provide an improvement over [79] where four different remote sensing datasets are considered for multi-class classification. This method is based on parameter isolation where two importance metrics - pixel affinity structure loss and representation consistency structure loss are calculated to preserve structural information across datasets. The method uses a shared network with addition of task-specific neurons in only the final layer.

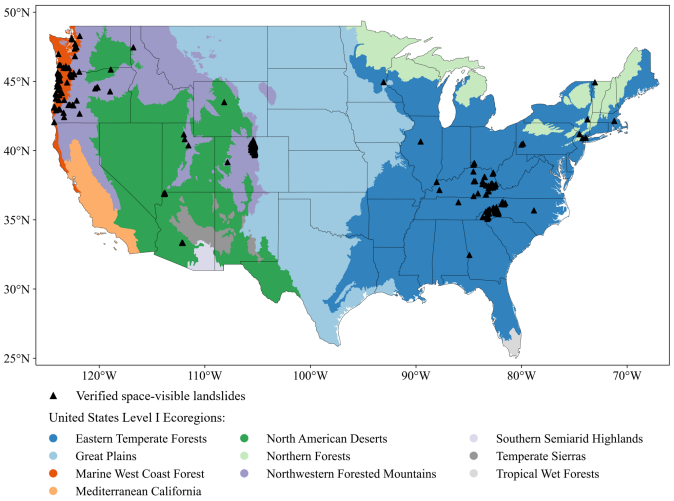

Fig. 2: **Study area of our dataset.** Distribution of identified space-visible landslides in the present study and Level I ecoregion map (https://www.epa.gov/eco-research/ecoregions-north-america) for the United States.

III. Methodology

In this section, we describe our methodology. Our dataset characteristics (multiple, visually dissimilar regions) provide the primary motivation for the design of TSMU. We describe our study area in Section III-A1 and dataset characteristics in Section III-A2. Data pre-processing steps are explained in Section III-A3. The proposed TSMU framework and network architecture are described in Section III-B. Since the architecture has shared parameters as well as ecoregion-specific parameters, we explain in Section III-C how the network determines the ecoregion of the input image in order to route the image through the network properly.

A. Dataset

The {latitude, longitude} coordinates for all landslide events in our study area and the subset of them we used in our experiments can be found on our GitHub link for data.

1) Study Area: We focused on space-visible landslides reported in multiple ecoregions in the US. These landslides have left visible scars and can be identified from satellite images. The occurrences of landslides are due to a combined effect of triggering mechanisms (e.g., rainfall and snow melt) and predisposing factors (e.g., hillslope environment, ecosystem dynamics, and geomorphic dynamics) [81], [82]. An ecoregion defines areas that share similar type, quality, and quantity of environmental resources, e.g., biomes and topography [83]. Due to the number of landslides we identified and the location of these landslides, we considered a relative coarse ecoregion level (i.e., Level I) to ensure each ecoregion had sufficient images. In the present study, the landslides came from the following ecoregions: Eastern Temperate Forests (ETF), Marine West Coast Forests (MWCF), North American Deserts (NAD), and Northwestern Forested Mountains (NWFM). Figure 2 shows the distribution of Level I ecoregions and the landslides identified in the present study.

2) Data Characteristics: Prior studies on landslide segmentation have used datasets with at least one of the following characteristics: (1) they contained few images, (2) the images were low-resolution, and/or (3) the images were collected from a small geographic location with homogeneous characteristics. For the purposes of this study, our goal was to collect a large quantity of sub-meter resolution landslide image pairs from geographically diverse regions.

The USGS provides a nationwide landslide inventory for the United States. The database contains more than three hundred thousand landslide point and polygon records and provides related information for the landslide event, such as the extent (when available) and times (when available) of landslide occurrence [7]. As the USGS landslide inventory is compiled from many different sources, the accuracy and quality vary between different landslide records. The USGS landslide inventory uses a semi-quantitative classification to rank the relative confidence in landslide occurrence and position for landslide records in their inventory. Five integer values are used to represent certainties of landslide records. These values are “1” (very low confidence), “2,” “3,” “5,” and “8” (very high confidence) [7]. Our data collection team only focused on landslide point records with event dates and a confidence level of at least five to obtain pre- and post-event landslide image pairs. Google Earth was used as the data source for the present study, which provides historical archives for multiple high-resolution aerial and satellite imagery. We examined available images in Google Earth around the landslide point location before and after the landslide date to obtain pre- and post-event images. Multiple pre- and post-event images were examined and saved as candidates. We filtered out satellite images with heavy cloud cover or low image quality; only the two highest quality pre- and post-event images with dates closest to the landslide event were chosen. Our data collection team visually examined and compared all the pre- and post-event image pairs to look for changes in land surface that are likely caused

TABLE I: Pixel Distribution Statistics

Ecoregion	# augmented training image pairs	Landslide Pixel Mean	Landslide Pixel Standard Deviation	Landslide Pixel Median	Non-Landslide Pixel Mean	Non-Landslide Pixel Standard Deviation	Non-Landslide Pixel Median	Mean Salience $\frac{\#\text{Landslide Pixels}}{\#\text{Non-Landslide Pixels}}$
Eastern Temperate Forests (ETF)	405	0.438	0.185	0.443	0.302	0.147	0.291	$\frac{5910254}{103665938} = 0.057$
Marine West Coast Forests (MWCF)	420	0.491	0.229	0.483	0.313	0.169	0.291	$\frac{13865231}{115633905} = 0.120$
North American Deserts (NAD)	429	0.612	0.217	0.654	0.464	0.182	0.469	$\frac{5093959}{110773689} = 0.045$
Northwestern Forested Mountains (NWFM)	400	0.417	0.184	0.419	0.316	0.148	0.314	$\frac{6879824}{96667056} = 0.071$

by landslides. Then these areas were manually annotated as landslides and used as ground truth for this study.

It should be noted that although the USGS landslide inventory contains a large number of landslide records, due to the availability of satellite images and the size of the landslide scars, we only identified 1,918 landslide points in the inventory from satellite images for the four ecoregions we considered in the present study. Among these landslide points, 112 points are located in ETF, 64 points are located in MWCF, 728 points are located in NAD, and 1,014 points are located in NWFM. A total of 496 georeferenced pre-event/post-event images pairs with an average size of 1200 by 800 pixels were collected for these identified landslide points in the four ecoregions (each post-event image may include more than one landslide point). The excel file containing the coordinates of these landslide events can be found on our GitHub link for data.

Landslides in the USGS inventory were curated by humans, and about two-thirds are in the three West Coast states [7]. This may be due to the use of high resolution DEMs from lidar data to include mapping of ancient deep seated landslides that are not triggered by individual rainfall events. Thus, not all of these may be detected by our approach. Also, our ability to identify landslides from satellite images varies from region to region due to differences in vegetation coverage, landslide size, and availability of satellite images. Thus, the data have an inherent domain imbalance in terms of the number of satellite images in each ecoregion: we obtained 27 images for ETF, 31 images for MWCF, 39 images for NAD, and 400 images for NWFM. To reduce the domain imbalance, images from ETF, MWCF, and NAD undergo data augmentation to increase the number of images in these ecoregions proportional to that of NWFM, thus obtaining 405, 420, and 429 images, respectively, as shown in Table I.

The landslide segmentation (pixel labeling) problem was formulated as a binary classification problem for each pixel ("is" versus "is not" a landslide pixel). We refer to the ground truth as a *binary mask*. A data annotation tool [84] was used to manually label the landslide areas in the satellite images. Data were annotated using images with original dimensions (before reshaping). The tool stored the labels as JSON files. These files were parsed to extract pixel-level information for generating binary masks.

Ground truth binary masks used for training were stored as gray-scale images with reshaped dimensions of 512 x 512 and pixel intensities {0,1}. Intensity values of zero represent non-landslide pixels (background) and intensity values of one represent landslide pixels (foreground).

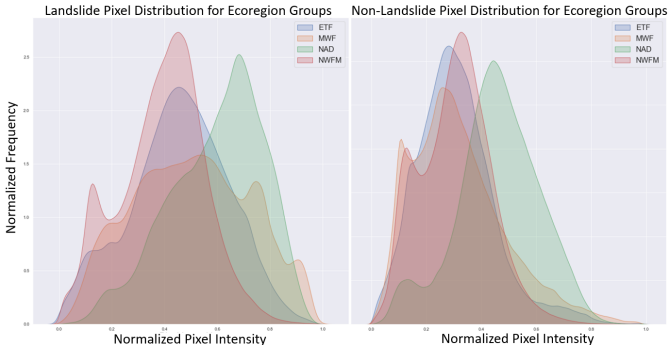

Fig. 3: **Histograms of landslide and non-landslide pixel intensities for the four ecoregions.**

Figure 3 shows the histogram of pixel intensities for landslide (foreground) and non-landslide (background) pixels for the different ecoregions. The corresponding quantitative statistics are shown in Table I. They indicate that each ecoregion, as well as landslides in each ecoregion, appear to have distinct characteristics. The mean salience indicates that NAD has the least number of landslide pixels, which indicate narrower landslides, followed by NWFM. ETF and MWCF have larger landslides.

More insight can be obtained by using t-stochastic neighbor embedding (t-SNE) [85], a tool for visualizing the similarities and differences in high-dimensional data. Figure 4 shows a two-dimensional t-SNE visualization of landslide pixels in the data. Each image is mapped to a single point in Figure 4, with similar images being mapped to points that are closer to each other. The points are colored by the ecoregion they come from, and an ellipse is drawn for each ecoregion to cover the bulk of the points from that region.

From the overlap in the ellipses in Figure 4, one can see a lot of similarities between the ecoregions; however, there are also a lot of differences (indicated by regions where the ellipses do not overlap). This observation motivates the network design choice of the proposed TSMU (described in Section III-B). Namely, the network has ecoregion-specific components, but

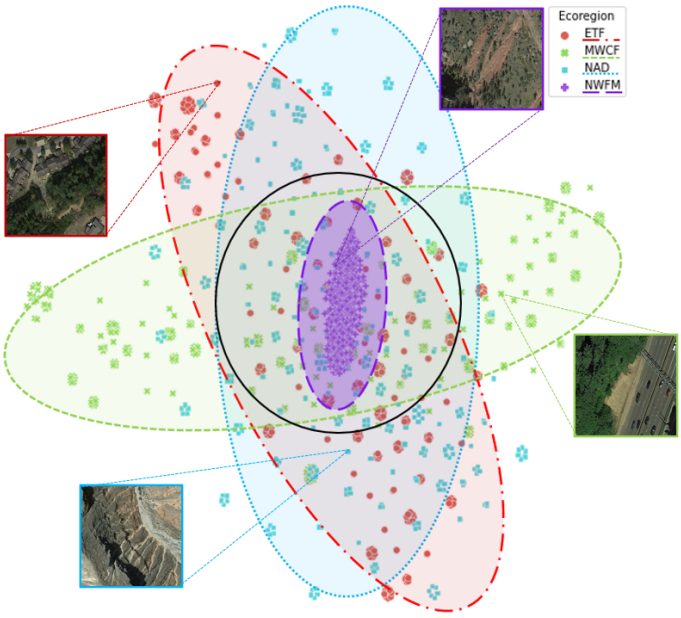

Fig. 4: **t-stochastic neighbor embedding (t-SNE) [85] visualization of landslide pixels of four ecoregions.** Each colored ellipse is a distribution of intensity of landslide pixels of an ecoregion, projected in two dimensions using t-SNE embedding. The black circle shows an overlap of all distributions, whereas distinct ecoregion-specific landslide pixels are also in the extremities of every ellipse. This data analysis provides the motivation for the design choice of TSMU. It can also be observed that NAD has more outliers than other ecoregions.

also a special subnetwork that is shared by all ecoregions.

Figure 4 also provides an intuitive explanation for our main experimental findings:

- The presence of substantial areas where the ellipses do not overlap is caused by properties that are unique to an ecoregion. Hence, a neural network trained on several ecoregions (or a small homogeneous region, as in prior studies) is unlikely to transfer well to new ecoregions because of this uniqueness.
- The presence of substantial areas of overlap mean that if we are interested in ecoregion R, it would be beneficial to collect data not just from ecoregion R, but also from other ecoregions to improve performance on ecoregion R. The reason is that images from these other ecoregions still contain useful information about landslides in ecoregion R. This finding has important implications for prior studies, which studied small homogeneous regions and were limited by the amount of data that can be collected from their study areas – their results could have likely been further improved by collecting data outside their study areas.

3) Data Processing: As a pre-processing step, satellite images were resized to 512×512 pixels, and normalized by dividing each pixel by the maximum intensity in the distribution. The final images used for training had pixel intensities in the range [0,1]. The changes in aspect ratios corresponding to each image were stored so that the images and their corresponding network predictions could be reverted to their original dimensions during the time of inference.

The dataset was used to form *training*, *validation*, and *test* sets as follows. The test set consisted of 5 (randomly selected) pre/post image pairs from *each* ecoregion. These images were *only* used for measuring the reported accuracy metrics. We use the term **global test set** to refer to the entire test set (20 images from four ecoregions) and highlight the contrast with an **ecoregion-specific test set** (which is the subset of the global test set belonging to the ecoregion of interest). After the testing images were removed from the data, a combination of random augmentations from the set of vertical flips, horizontal flips, and rotations were performed on the remaining images, resulting in the training set as shown in Table I.

We also created a validation set for tuning hyperparameters such as the learning rate and momentum parameter (β) of our optimizer (Adam [86]) and it was never used to report accuracy metrics (i.e., the validation set is disjoint from the testing set). We constructed the validation set from the *training data* by applying multiple augmentations from the set of the following random augmentations to each image pair: diagonal flip, horizontal flip, vertical flip, zoom, translation, and Gaussian blur. We note that this is a larger collection of augmentations than is applied to the training set. This was done intentionally so that the validation set has slightly different characteristics from the training set, thus reducing the chance of the model overfitting to the training set.

B. Task Specific Model Updates (Proposed Model)

In this paper, ecoregions and "*tasks*" are used interchangeably. The biggest challenge in building a semi-automated global landslide database is how to update the landslide segmentation model as images from new ecoregions are acquired. Hence, there is a need for a semantic segmentation model that has the ability to learn new tasks while retaining the same performance on old tasks. We consider the following idea: because landslides in different ecoregions share similarities as well as differences, a model should have components/parameters that are shared by all ecoregions, as well as ecoregion-specific components. With this in mind, we propose *Task-Specific Model Updates* (TSMU), an extension of *Learning without Forgetting (LwF)* [17] from image classification to semantic segmentation. This extension requires the network architecture to use a so-called encoder/decoder model, using loss functions more appropriate to semantic segmentation, and a multi-stage training procedure that becomes necessary because of the increased network complexity (because semantic segmentation is a much more complex task than image classification).

A final distinction is that at deployment time (after training has finished), the ecoregion of an input image is not known and must be inferred by the network, while in the LwF framework, it would be assumed to be known. This is because each ecoregion is a "task" and LwF assumes that the task of interest is known at deployment time.

An overview of TSMU is shown in Figure 5, with ecoregion inference discussed in Section III-C.

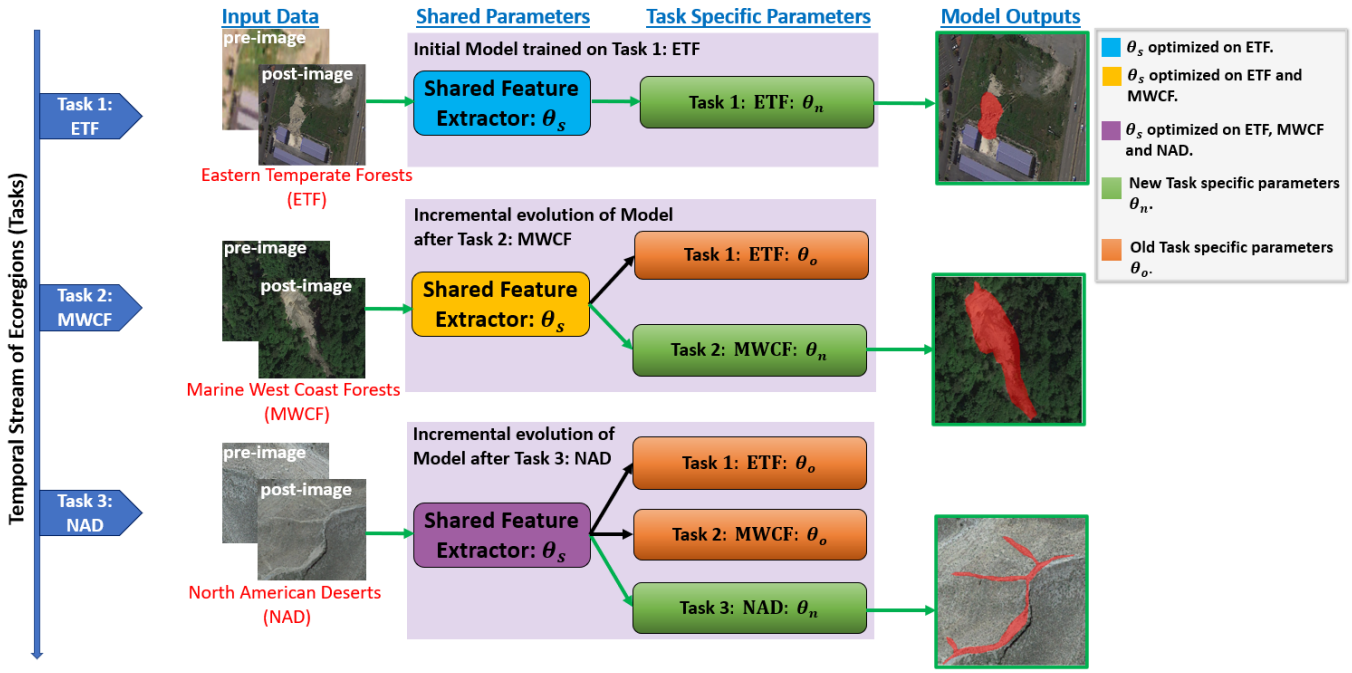

Fig. 5: **Illustration of proposed TSMU**, illustrated for data from 3 ecoregions acquired sequentially. The semantic segmentation network contains a shared encoder (Figure 6) that extracts features from images and also contains multiple decoders, one for each ecoregion (task). Each decoder outputs a landslide segmentation (i.e., "decodes" features into a landslide annotation), and the ecoregion determines which decoder's output is used. The encoder follows the ResNet-34 architecture [43] while the decoders use the U-Net architecture (Figure 7) [41]. When a new ecoregion (new task) is encountered, instead of creating a completely new model, TSMU adds a new decoder to the existing model. The goal of TSMU is to update the network parameters in the new decoder *and* the shared encoder to (1) achieve acceptable accuracy on the new ecoregion (new task) while (2) maintaining or improving performance on the old ecoregions. This image is best seen in color.

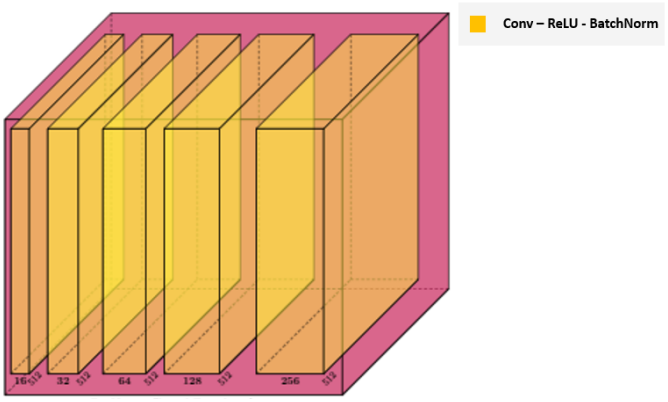

Fig. 6: **ResNet-34 Shared Feature Extractor:** θ_s This figure shows the ResNet-34 encoder that serves as the shared feature extractor for TSMU. The model consists of sequential {Conv-ReLU-BatchNorm} blocks. The spatial resolution remains 512 throughout, while the depth of the activations increases from 16 in the first layer to 256 in the last layer. The output of the shared encoder is the input to the U-Net Decoder.

1) Problem Definition: We formally define the problem as follows. Consider a sequence of semantic segmentation tasks for landslide mapping, denoted by $T = \{T_1, T_2, T_3, ..., T_t, ...\}$. Each task T_t is the arrival of a new batch of data from ecoregion$_t$ and when a new task is encountered, new task-specific parameters are added to the network. Each task T_t has training data $D_t = \{x_i^t, y_i^t\}|_{i=1}^{n_t}$ with n_t number of samples. In our case, x_i^t is the i^{th} input image pair (a pre- and a post-landslide image) of Task t and y_i^t is the corresponding ground truth mask (i.e., labels for each pixel in the image). The set of all data up to task t is denoted by $D_{[1:t]}$.

The shared parameters (shared by all ecoregions) are denoted by θ_s, and the specific values they take after the network is updated (via training) with task T_t are denoted by $\theta_s^{(t)}$. Similarly, $\theta_1, \theta_2, \ldots, \theta_t$ are the task-specific parameters for tasks $T_1, \ldots, T_t$. The values of the parameters θ_i after the network is updated (via training) for task T_t are denoted by $\theta_i^{(t)}$. We let $\Theta_t = \{\theta_s^{(t)}, \cup_{i=1}^t \theta_i^{(t)}\}$ to be the values of all parameters after the network is updated with task t. In the continual learning paradigm, when task T_t is being added, only data D_t for the current task (ecoregion) are available and prior data (i.e., $D_{[1:t-1]}$) are not accessible. The goal is to train the model on the new data while preserving or improving performance on the prior tasks, so that the model does not undergo catastrophic forgetting.

The pipeline of TSMU is shown in Figure 5. Initially, there is one task (e.g., the ETF ecoregion) and the initial network is trained with parameters $\theta_s^{(1)}$ (parameters to be shared with future tasks) and $\theta_1^{(1)}$ (parameters for Task 1). When data from the MWCF ecoregion are collected, an additional set of parameters are added and all of the parameters are updated, resulting in $\theta_s^{(2)}, \theta_1^{(2)}$ and $\theta_2^{(2)}$. The reason that parameters for task T_1 are updated when data from Task T_2 arrive is that the

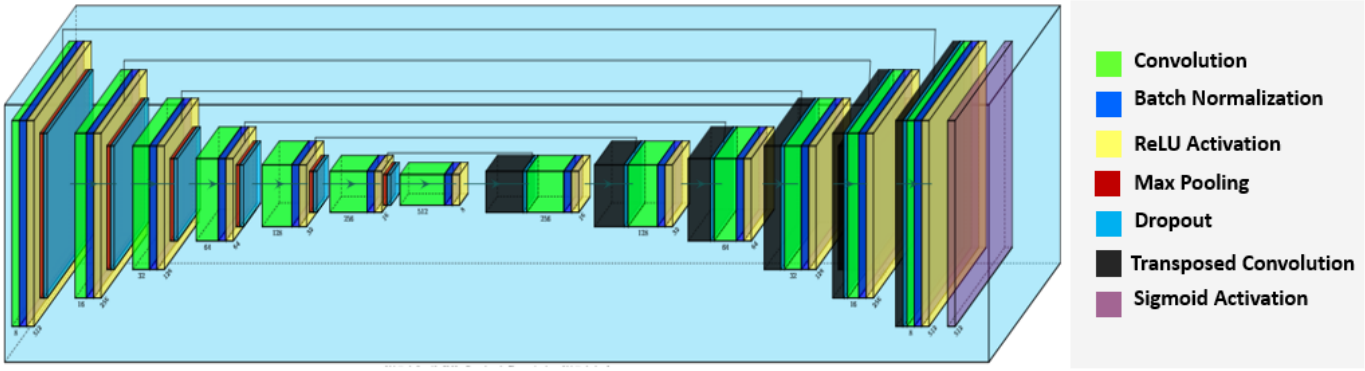

Fig. 7: **U-Net Task Specific Decoder Architecture.** This figure shows the U-Net architecture that serves as task specific parameters (θ_o and θ_n) for TSMU. The model consists of an an encoder made up of {Conv-ReLU-BatchNorm-MaxPool-Dropout} layers, which reduces the spatial resolution from 512 to 8. This is followed by a decoder made up of {Transposed Conv-Dropout-Conv-BatchNorm-ReLU} layers, which increases the spatial resolution back to 512. A sigmoid activation is used to get the output probability map.

shared parameters θ_s get updated (from $\theta_s^{(1)}$ to $\theta_s^{(2)}$) and so θ_1 must be updated to make sure that the changes in θ_s do not cause a performance degradation on task T_1. When task T_3 arrives, then new parameters are added and all parameters up to this point are updated, etc.

2) Base Network Architecture: In order to accomodate shared and task-specific parameters, our model takes the form of a general Encoder-Decoder [87] architecture. The Encoder converts a pair of pre-landslide/post-landslide images into a "feature representation" and is shared among all ecoregions (tasks). The Decoder takes the feature representation and creates pixel-wise predictions from it. The way that features are converted into predictions is ecoregion-specific, and so each Decoder corresponds to an ecoregion and its parameters are the task-specific parameters. The input dimensions are $512 \times 512 \times 6$ (each image in a pair is $512 \times 512 \times 3$ having 3 channels: {R, G, B}). The base architectures of shared encoder and task specific decoder we used for these experiments is illustrated in Figure 6 and Figure 7, respectively.

The encoder (Figure 6) is a subset of the ResNet-34 [43] architecture. We let θ_s denote the parameters of this encoder. The convolutional layers [88] of this encoder start with a depth of eight 3x3 convolutional filters and end at 256 filters. Thus the output of the encoder has dimensions $512 \times 512 \times 256$.

The decoder (Figure 7) is a variant of the U-Net architecture [41]. It consists of a contracting path (first half of the blue block of Figure 7) followed by an expansive path (second half of the blue block of Figure 7). The contracting path is used by the network for additional (ecoregion-specific) feature engineering and the expansive path uses these features to label pixels. The reason for this architectural choice is that it will easily allow us to extend the network for new ecoregions (the ResNet encoder will be shared by all ecoregions and will learn how to extract features common to all ecoregions; each new ecoregion will then be explicitly learned by a separate U-Net decoder that will further extract region-specific representations in the contracting path, followed by pixel labeling in the expansive path.

In the decoder, a repeating building block R_C is used in the contracting path. It consists of a 3×3 separable convolution (to reduce the number of trainable parameters) [89], [90] and a batch normalization layer [91], followed by a rectified linear unit (ReLU) [92] activation function. Seven such R_C blocks are used, starting with a depth of 8 filters (channels) and ending with 512 filters. A 2×2 max pooling [93] layer with a stride of 2 is used between each R_C block to downsample the resolution by half. The number of filters is doubled at every down-sampling step. This is followed by a spatial dropout [94] layer acting as a regularizer to avoid overfitting. For the expansive path in the decoder, we use a repeating block R_E. It consists of a 2×2 upsampling block that uses a transposed convolution (deconvolution) [95] layer. This upsamples the layer's input as well as decreases the number of channels by a factor of two. A skip connection [96] from the contracting path at every downsampling step is concatenated with the corresponding output of the deconvolution block to get back the pre-upsampled resolution. This is fed into into a 3×3 convolution followed by a Batch Normalization layer and a ReLU activation. Six such R_E blocks are used, starting with a depth of 256 filters and ending with 8 filters. At the final layer, a 1×1 convolution [88] with sigmoid activation is used to generate predictions at the same spatial resolution as that of the input image. The final output from the decoder is of resolution $512 \times 512 \times 1$.

3) Training Algorithm: The algorithm for TSMU is outlined in Algorithm 1. Given the complete set of parameters Θ_t up to task t, the prediction of the model on input x is denoted as $f(x \mid \Theta_t)$. Note that this prediction is a matrix. Each entry of the matrix is a prediction for the corresponding pixel and represents the predicted probability that the pixel is a landslide pixel. As we add decoders to the model, we will use subscripts like f_j to refer to the output of decoder j. Because data $D_{[1:t-1]}$ are unavailable for task t, the idea is to use all the previously learned parameters Θ_{t-1} (up to the previous task $t-1$) to preserve the model's performance on old tasks. This is the key idea from LwF that we are using. Specifically, we feed the input from the new ecoregion into the network and record the responses of the decoders for all

ecoregions. The goal of the parameter updating is to improve performance on the new ecoregion while trying to keep the responses of the other decoders on this input from changing. Formally, the training approach is as follows:

A **Initial Training**: In this step, we train the base architecture of our model described in section III-B2 for the first task T_1 consisting of data $D_1 = \{x_i, y_i\}|_{i=1}^{n_1}$. The data D_1 can consist of image pairs from one or more ecoregions. The values of the parameters after training the model on D_1 is $\Theta_1 = \{\theta_s^{(1)}, \theta_1^{(1)}\}$, where θ_s represents the common encoder (Figure 6) that will be shared with future ecoregions and θ_1 represents the first task specific decoder (Figure 7). The parameter set is optimized until convergence on data D_1 using Soft Dice Loss L_{SD}^1 [97] with Adam [98] optimizer. The output of decoder 1 for an input x is denoted by $f_1(x|\Theta_1)$.

B **Incremental Freeze Training**: Now data from a new task (ecoregion) come in and we must update the network. This is done in two steps, incremental freeze training and joint fine-tuning. We describe incremental freeze training step first. Without loss of generality, let T_t be this latest task and D_t be the corresponding data. In this case, the most recently updated model currently has parameters Θ_{t-1}. The output of decoder j (for $j = 1, \ldots t-1$) on an input x would be denoted as $f_j(x|\Theta_{t-1})$. We add a new decoder for this new ecoregion and randomly initialize its task specific parameters θ_t. We freeze the weights Θ_{t-1} (which are the shared parameters and parameters of the previous $t-1$ tasks) and only train the new weights θ_t with the new data. Soft Dice Loss L_{SD}^t with Adam optimizer is used for optimization. The resulting values of the parmeters θ_t are denoted as θ_t^*.

C **Joint Fine Tuning**: Next step, we jointly fine tune *all* parameters using Knowledge Distillation loss (KD) [54]. For each input x_i^t in the new data, we run it through each of the decoders of the previous tasks and store the results. That is, for $j = 1, \ldots, t-1$ we let $z_{i,j} = f_j(x_i^t|\{\Theta_{t-1}, \theta_t^*\})$ denote the stored response of decoder j for input x_i^t. Then we train all of the parameters together using a loss function that consists of two parts. The first part of the loss function, denoted by L_{KD}^t uses knowledge distillation loss so that the output of decoder j on input x_i^t would be close to $z_{i,j}$ for all of the prior tasks ($j = 1, \ldots, t-1$). Thus, even as the shared encoder parameters θ_s change, the parameters of the old task decoders are updated to compensate for this change so that their output for x_i^t remains almost the same. Meanwhile, we use soft dice loss, denoted by L_{SD}^t to try to force decoder t (of the actual ecoregion of the input) to try to match the ground truth y_i^t for input x_i^t. The overall loss is a weighted sum $\lambda_1 L_{SD}^t + \lambda_2 L_{KD}^t$, where λ_1 and λ_2 are hyperparameters that are tuned on the validation set using a grid search. We also add weight decay regularization [99] during this training step. Once this training step is done, the updated parameters are now referred to as $\Theta_t = \{\theta_s^{(t)}, \cup_{i=1}^t \theta_i^{(t)}\}$.
Now, as each new ecoregion is added, a corresponding decoder for the ecoregion is added and the incremental freeze training and joint fine-tuning steps are applied to update the network parameters.

Algorithm 1 Task-Specific Model Updates Training

Input:
Sequential Semantic Segmentation Tasks: $T = (T_1, T_2, ..T_t...)$
Training Data: $D_t = \{x_i^t, y_i^t\}|_{i=1}^{n_t}$ with n_t samples
Hyperparameters for Loss functions: $\lambda_1, \lambda_2, \mu_o, \tau$
Output:
Parameter Set optimized for Task T_t: $\mathbf{\Theta_t} = \{\theta_\mathbf{s}^{(\mathbf{t})}, \cup_{\mathbf{i=1}}^{\mathbf{t}} \theta_\mathbf{i}^{(\mathbf{t})}\}$

while Task T_t **do**
  Get Training Data D_t for the current task
  if t = 1 **then** ▷ First Task
    Initial Training:
    Randomly initialize Parameter Set $\Theta_1 = \{\theta_s, \theta_1\}$
    Output of decoder 1 on input x is $f_1(x|\Theta_1)$
    Compute Soft Dice Loss L_{SD}^1 ▷ Eq 3
    Update parameter set Θ_1 with Adam optimizer:
      $\Theta_1 = \{\theta_s^{(1)}, \theta_1^{(1)}\} = \text{argmin}_{\{\theta_s^k, \theta_1^k\}} L_{SD}^1$
  else ▷ $t = \{2, 3, ...\}$
    Incremental Freeze Training:
    Add new decoder j for data D_t
    Randomly initialize its task specific parameters θ_t
    Freeze values of: $\Theta_{t-1} = \{\theta_s^{(t-1)}, \cup_{i=1}^{t-1} \theta_i^{(t-1)}\}$
    Output of new decoder j on input x is $f_j(x|\theta_t)$
    Compute Soft Dice Loss L_{SD}^t ▷ Eq 3
    Update new parameter set θ_t with Adam optimizer:
      $\theta_t^* = \text{argmin}_{\{\theta_t^k\}} L_{SD}^t$
    Joint Fine Tuning:
    Make all parameters of Θ_{t-1} trainable ▷ Unfreeze
    Store $z_{i,j} = f_j(x_i^t|\{\Theta_{t-1}, \theta_t^*\})$, $j = 1, ..t-1$
    Compute $f_j(x_i^t|\Theta_t)$, $j = 1, ..t-1$
    Compute $f_t(x_i^t|\Theta_t)$
    Compute Soft Dice Loss L_{SD}^t ▷ Eq 3
    Compute KD Loss L_{KD}^t ▷ Eq 5
    Compute Total Loss
      $L_{total}^t = \lambda_1 L_{SD}^t + \lambda_2 L_{KD}^t$
    Update parameter set Θ_t:
    $\theta_s^{(t)} = \text{argmin}_{\{\theta_s^k\}} L_{total}^t$
    $\cup_{i=1}^{t-1} \theta_i^{(t)} = \text{argmin}_{\{\theta_i^k\}} L_{total}^t$
    $\theta_t^{(t)} = \text{argmin}_{\{\theta_t^k\}} L_{total}^t$
return output $\Theta_t = \{\theta_s^{(t)}, \cup_{i=1}^{t-1} \theta_i^{(t)}, \theta_t^{(t)}\}$

4) Loss Functions: Soft Dice Loss L_{SD}^t is a differential approximation of the Intersection Over Union (IoU) metric (IoU measures the intersection between the predicted landslide pixels and actual landslide pixels, then divides by their union). We use L_{SD}^t to force decoder t to try to match the ground truth y_i^t for each input x_i^t in the new data at time step t. The output of decoder t is denoted by $f_t(\cdot)$. The Intersection I^t and Union U^t are given by

$$I^t = \left|\left| \sum_{i=1}^{n_t} y_i^t \otimes f_t(x_i^t|\Theta_t) \right|\right|_1 \quad (1)$$

$$U^t = \left|\left| \sum_{i=1}^{n_t} y_i^t + f_t(x_i^t|\Theta_t) \right|\right|_1 \quad (2)$$

where $\otimes$ denotes the pointwise product (recall that the mask and predictions are matrices with an entry for each pixel) and the L_1 norm sums up over the pixels (matrix entries).

The Soft Dice loss L_{SD}^t, with hyperparameter μ_0, is:

$$L_{SD}^t = 1 - \frac{2I^t + \mu_o}{U^t + \mu_o} \quad (3)$$

The value of μ_0 was set to 1 based on a grid search on the validation set.

At the end of Incremental Freeze Training step, the resulting values of parameters θ_t of the new decoder t are denoted by θ_t^*. For each input x_i^t in the new data at time step t, we evaluate and store the outputs of each of the decoders of the previous tasks $T_{[1:t-1]}$. For decoder j, where $j = 1, ..., t-1$, the stored response for input x_i^t is given by

$$z_{i,j} = f_j(x_i^t|\{\Theta_{t-1}, \theta_t^*\}) \quad (4)$$

The Knowledge Distillation loss L_{KD}^t is given by

$$L_{KD}^t = \left|\left| -\sum_{j=1}^{t-1}\sum_{i=1}^{n_t} \Phi(z_{i,j}) \otimes \log(\Phi(f_j(x_i^t|\Theta_t))) \right|\right|_1 \quad (5)$$

where the log is taken pointwise, $\otimes$ is pointwise multiplication, and $\Phi(\cdot)$ is the rescaling function (also applied pointwise) that has a hyperparameter τ:

$$\Phi(\mu) = \frac{\mu^{1/\tau}}{\mu^{1/\tau} + (1-\mu)^{1/\tau}} \quad (6)$$

Here τ was set to 2 based on a grid search on the validation set. As before, the L_1 norm in Equation 5 adds up the values (they are always positive) over the pixels.

C. Selecting Decoders at Test Time

At deployment time, a new pre/post image pair is provided, with the goal of identifying if and where landslides occurred between the date of the pre-image and the date of the post-image. Pre/post image pairs can be collected automatically and so are typically not labelled with the ecoregion they belong to. Thus, we propose an automated method that determines which ecoregion a new image pair belongs to and hence which decoder of TSMU to use for landslide labeling. We train an *autoencoder* [100] gate for each ecoregion separately, as shown in Figure 8. An autoencoder is a neural network that learns to compress its input (nonlinear dimensionality reduction) and then decompress it (lossy reconstruction). Each autoencoder gate A_t is trained until convergence using binary cross entropy loss [101] with Adam optimizer to minimize the reconstruction error e_t. After learning the autoencoders $\{A_t\}_{t=1}^{\tau}$ for all the tasks, a softmax layer is added to take the reconstruction errors $\{e_t\}_{t=1}^{\tau}$ as inputs, given a test sample x. The reconstruction error e_t of the t-th autoencoder A_t is the Euclidean distance between output of A_t and x. The softmax layer gives the confidence values of each autoencoder. The most confident autoencoder indicates the corresponding

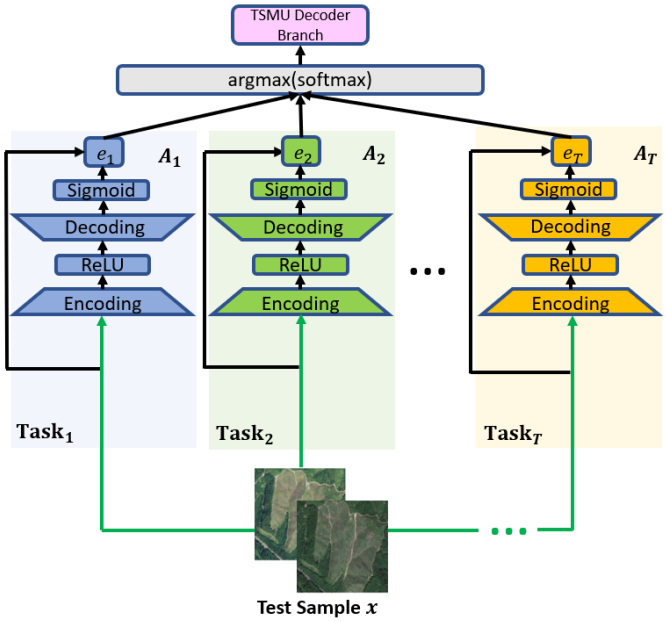

Fig. 8: **Test-time TSMU decoder selection using autoencoder gates.** This figure shows the mechanism used to select the decoder branch for predicting the output probability map of an unknown test sample x.

decoder of TSMU to choose for test-time prediction of sample x. The probability p_i of autoencoder A_i is given by:

$$p_i = \frac{e^{(-e_i/\tau)}}{\sum_{i=1}^{T} e^{(-e_i/\tau)}} \quad (7)$$

The hyperparameter τ was set to 2 using grid search on the validation set. For tasks that have some overlap, it can be convenient to activate more than one decoders and take the average map of the output probabilities.

IV. Experiments & Results

In this section, we provide a detailed description of our experimental setup and the results. The implementation details and evaluation metrics are described in Sections IV-A and IV-B, respectively. Our experiments have two main variants (*spatial holdout* and *sequential*) that we explain in Section IV-C. In Section IV-D, we describe the baselines and competing methods we use for comparisons. In section IV-E, we present the quantitative experimental results and in section IV-F, we present some qualitative case studies.

A. Implementation Details

All of our models were trained with an initial learning rate of $1e^{-3}$ for 10k iterations with a batch size of 2 on an NVIDIA RTX 2080 Ti GPU cluster with 16GB memory. We use a decay of 0.001 and momentum of 0.9 for the Adam optimizer. We use a learning rate scheduler to exponentially decay the learning rate by a factor of 0.01 until we reach $1e^{-8}$. Data preprocessing and the creation of training, validation, and test sets were performed as described in section III-A3. We average the results by performing all the experiments three times to decrease the effect of randomness during training.

B. Evaluation Metrics

All our models take as input a pair of pre-event and post-event (bi-temporal) images and, for each pixel, output a number between 0 and 1 (which is interpreted as a probabilistic estimate that the pixel is part of a landslide). We convert each per-pixel probability into a per-pixel prediction value of 0 or 1 (1 = landslide and 0 = background) by thresholding probabilities at 0.5.

For quantitative evaluation of the models, we use mean intersection over union [102] and the following continual learning metrics from the literature that test the amount of forgetting that a model may experience: Average Continual Accuracy (ACA) [80] and Average Forgetting (AF) [62]. All of these are defined next.

Mean intersection over untion (**mIoU**) is the area of overlap between predicted and ground truth pixels divided by the area of their union, averaged over all the classes, the classes being "landslide" and "background" (non-landslide). The mIoU of a model trained on sequential tasks T_1 to T_k calculated on the test set of task T_j can be mathematically written as:

$$\text{mIoU}_{k,j} = \frac{1}{2}\sum_{x=0}^{1}\frac{p_{xx}}{\sum_{y=0}^{1}[p_{xy}+p_{yx}]-p_{xx}} \tag{8}$$

where p_{xy} denotes the number of pixels of category x predicted as category y. mIoU is evaluated over the testing data.

Let w_j be the number of test images in Task T_j (in our case, the number of testing images from each ecoregion is the same, as explained in Section III-A3). After the model has been updated with the kth task, the Weighted Average (WA) of mIoU on all tasks from T_1 up to task T_k can be written as

$$\text{WA}_k = \frac{1}{\sum_{j=1}^{k} w_j}\sum_{j=1}^{k} w_j \times \text{mIoU}_{k,j} \tag{9}$$

The Weighted Forgetting (WF) up to task T_k is defined by considering, for each task, the difference between the mIoU of the best prior model for that task and the current model, and then averaging across tasks. Mathematically it can be written as:

$$\text{WF}_k = \frac{1}{\sum_{j=1}^{k-1} w_j}\sum_{j=1}^{k-1} w_j \times f_j^k \tag{10}$$

where

$$f_j^k = \max_{\{l\in\{1,..k-1\}\}}\{\text{mIoU}_{l,j}\} - \text{mIoU}_{k,j} \quad \forall j < k \tag{11}$$

Positive values of WF_k mean that the most recent model's performance has degraded for the prior tasks, while negative values of WF_k mean that the current model's performance on the prior tasks is better than before.

C. Experimental variants: spatial holdout vs. sequential

We consider the setting where data arrive over time in batches, and every new batch of data may belong to a different ecoregion with distinct landscape characteristics. We perform our experiments with two variants of this data collection setting, which we call *spatial holdout* and *sequential*. (1) **Spatial Holdout**: This setting is designed to test how well a *single* update works. Thus, in this setting, data from three ecoregions are available in the beginning and data from one ecoregion will arrive later. Note that since we have 4 total ecoregions, there are 4 possible ways of doing this grouping (and we consider all of them). The initial group of 3 ecoregions is treated as one task (i.e., they all share the same encoder and decoder) and the 4th ecoregion, which arrives later in time, will get its own task-specific decoder in the TSMU framework. (2) **Sequential**: This setting considers how well *multiple* sequential updates are performed. Initially, data points from only one ecoregion are available. Then, data from the rest of the ecoregions arrive, one ecoregion at a time. After each ecoregion, a new task-specific decoder is added and a model update is performed. Hence, our final model will have four task specific decoders at the end (Figure 7).

D. Compared Methods

1) Baseline Catastrophic Forgetting Experiments: We perform six incrementally complex baseline experiments for comparison with TSMU and highlight the value of each added procedure. The first four methods do not involve revisiting old data, while the last two revisit old data (hence they serve to measure what performance, if any, is lost when operating in the more restricted setting that TSMU was designed for). All the baseline experiments use the spatial holdout (IV-C) data collection setting to explore the performance of TSMU when one new ecoregion is added.

1) **Naive Transfer (NT) [103]**: The first reference option for a baseline, is to simply not update the network in response to the new training data. This allows us to evaluate how well a model trained on one homogeneous region performs on other regions, and to determine whether adding training data from those regions is necessary. We use this for the spatial holdout experiments, so the base architecture described in Section III-B2 is trained on three ecoregions at a time, which serves as the old task. The new task is the arrival of data from the fourth ecoregion.
2) **Retraining [18]**: The next option is to take a previously trained model and continue training it using data from only the new ecoregion. In order to guard against catastrophic forgetting, this subsequent training is done using a small learning rate, which is a common DL practice. The purpose of this baseline is to explore performance when no new parameters are added in response to data from the new ecoregion.
3) **Feature Extraction [75]**: The next option is to add new task specific parameters (Figure 7) in response to data from new ecoregions. However, the shared parameters of the encoder (Figure 6) are frozen for the new data (i.e., they are not allowed to change) and hence their purpose is to extract features from the new ecoregion using the knowledge of how to extract features obtained from the initial task. The decoder (i.e., task-specific weights) that is added to the model has weights that are trainable. This method avoids catastrophic forgetting as the response of the model to old data remains unchanged (because the

shared parameters do not change and the decoders for the previous tasks also do not change).

4) **Fine Tuning [74]**: This is an extension of Feature Extraction. In this option, unlike Feature Extraction, the shared parameters along with the newly added task specific parameters are trained together. This changes the response of the model to old data due to the change in the shared parameters and so has the potential to cause catastrophic forgetting. The purpose of this baseline is to explore the performance when new task specific parameters are added and the shared parameters change in response to new data.
5) **Joint Training with all data (JTAD)** This is the approach of training the entire TSMU architecture (shared encoder and region-specific decoders) using *all* of the data collected at a given time step (i.e., old data points are revisited). This allows us to measure what performance may be lost in the more restrictive setting that TSMU is forced to work in (with data arriving over time, and without the ability to revisit old data). JTAD is the most computationally expensive baseline and is expected to be the most accurate. Thus, the goal of TSMU is to approach the accuracy of JTAD despite operating in a more restricted setting.
6) **Joint Training with all data - Single Decoder (JTAD-SD)**: This approach, like JTAD, uses all of the data at once, but only uses a single decoder for all regions. The purpose of this baseline is to measure the effect of using region-specific parameters (JTAD vs. JTAD-SD).

2) Competing methods in Continual Learning: We also compare TSMU framework with several state-of-the-art continual learning mechanisms to evaluate its effectiveness of alleviating catastrophic forgetting. We consider the following three competitive methods for continual learning for segmentation. All of them use variants of Knowledge Distillation [54] in their objective functions. (1) KD-BCE [79] is a regularization-based continual learning strategy that performs multi-class remote sensing semantic segmentation. This method uses a replay based mechanism to sample data from old classes. The objective function in this method penalizes the shift of important parameters in the network. However, the method assumes a class continual learning setting where the new class and the old class data come from the same underlying data distribution. (2) Incremental Learning Techniques for Segmentation (ILT) [104] is a parameter isolation based method that proposed four KD variants, out of which we choose the one they reported works the best (output feature alignment distillation loss implementation). This method again assumes class continual mechanism where a class is added sequentially while the data arrive from the same underlying distribution. (3) Continual learning with Structured Inheritance (SI) [80] is another parameter isolation based method, which in addition to KD, introduces two more terms in the objective function: Pixel Affinity Structure (PAS) loss and Representation Consistency Structure (RCS) loss. The paper works with four remote sensing datasets. Their approach is class continual as well, where new classes from a different dataset are added sequentially to the same network. Their datasets are collected from similar geographical locations with multiple class overlap. This means that the sequentially added datasets the paper uses do not undergo substantial domain shift. This is the main reason their experiments do not need task specific parameters. Unlike the three methods, TSMU is a regularization based network expansion method, where we add task specific parameters for every newly encountered ecoregion.

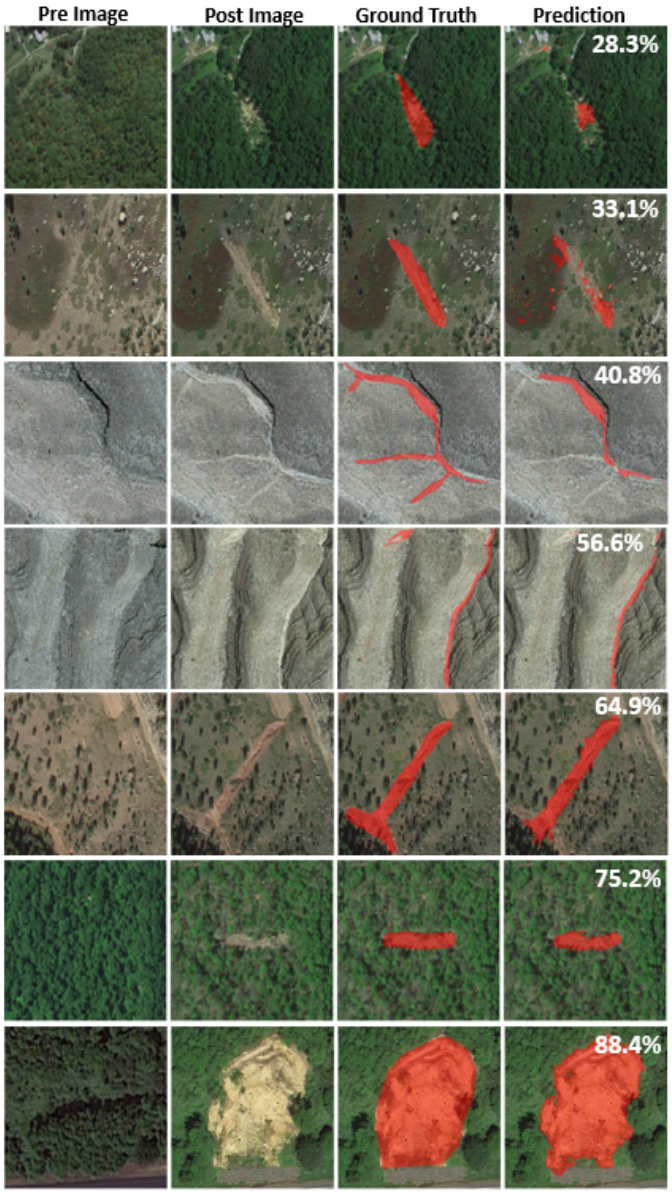

Fig. 9: **Sample images to help interpret mIoU scores.** Sample images containing (from left to right) the pre-event, post-event, human-generated landslide annotations ("ground truth"), and predictions of various quality, with mIoU scores ranging from 28.3% to 88.4%. The purpose is to help understand how mIoU is correlated with visual quality. It can be observed that the number of false positives and false negatives decrease as the mIoU score increases. mIoU values around 65% are considered fairly good. The excel file containing the coordinates of the landslides used in this figure will be available in our GitHub link for data.

E. Results

In this section, we quantitatively compare TSMU framework with baseline and state-of-the-art methods under two data collection settings. These comparisons also inform us of the effect of each added operation. Human-generated pixel-wise landslide annotated binary masks are treated as the ground truth for all experiments. The intensity at each pixel is either 0 or 1, where 0 represents non-landslide pixels and 1 represents landslide pixels. A sample binary map can be seen in Figure 1.

1) Spatial Holdout: These experiments use the spatial holdout (IV-C) data collection setting. All metrics are computed over the test images.

(i) Model Transferability:

TABLE II:
PERFORMANCE COMPARISON (MIOU) FOR NAIVE TRANSFER BASELINE. THE MODEL IS TRAINED ON THREE ECOREGIONS (OLD TASK) AND EVALUATED ON (1) TESTING DATA FROM THOSE ECOREGIONS AND (2) TESTING DATA FROM A NEW ECOREGION (NEW TASK).

Naive Transfer Baseline			
Data: Old Task	mIoU: Old Task	Data: New Task	mIoU: New Task
ETF + MWCF + NAD	66.6	NWFM	38.5
MWCF + NAD + NWFM	67.4	ETF	47.4
NAD + NWFM + ETF	68.8	MWCF	44.2
NWFM + ETF + MWCF	69.1	NAD	21.8
Average mIoU:	**67.9**		**37.9**

The Naive Transfer experiments, presented in Table II show that models trained on an initial set of ecoregions do not transfer very well to new ecoregions. It can be observed from Table II that all models have a pixel-level mIoU in the high 60s when evaluated on the testing data of the initial set of ecoregions [80], but have drastically lower mIoU on the new ecoregion. A pixel-level mIoU of above 60 indicates fairly good performance. To understand how mIoU is related to visual prediction quality, Figure 9 shows some sample mIoU scores along with ground truth and predicted landslide pixels.

We note from Table II that NT suffers a severe decline in performance when used for a new ecoregion. The declines are larger than 20%, with the average at 30% (from 67.9% to 37.9%). The poorest transferability is found with the model trained on ETF + MWCF + NWFM with a mIoU of 69.1 is tested on NAD, resulting in mIoU of 21.8. This is because NAD has substantially narrower landslides with very different background pixel distribution as shown in Section III-A2. The learned representations of the trained model seems unable to generalize on a new unseen task, which is characteristically dissimilar compared to the data used to train the model. This experiment also implicitly indicates the domain shift in ecoregions as the cause for the drastic drop in mIoU. This experiment highlights the geographic specificity of landslide detection and that training only over homogeneous regions is insufficient for building a large-scale/global landslide identification model.

(ii) Model Updates:

Having established that models do need to be updated with training data from new ecoregions, we now compare, in Tables III to VI, the competing model update methods described in section IV-D. Here we are still in the spatial holdout setting, in which a network is trained on an initial set of three ecoregions and then a model update is performed using training data from the fourth region. In the tables, we use *Ref mIoU (old)* to refer to the mIoU, on the old task, of Naive Transfer. *Ref mIoU (new)* refers to the mIoU of Naive Transfer when evaluated on the test set of the new ecoregion. These reference mIoUs are used as baselines to determine the amount of forgetting on the old task and amount of improvement on the new task when the model update methods are applied.

Tables III to VI feature the same experimental setup, except the difference in which ecoregion is treated as the new task.

TABLE III:
PERFORMANCE COMPARISON (MIOU) FOR MODEL UPDATE METHODS STARTING FROM {ETF+MWCF+NAD} TO {NWFM}

Naive Transfer baseline: Ref mIoU (old) = 66.6 ; Ref mIoU (new) = 38.5			
Methods	[ETF + MWCF + NAD] (old) → [NWFM] (new)		Weighted Average (WA)
Retraining [18]	37.4	65.9	44.5
Feature Extraction [75]	66.6	48.8	62.7
Fine Tuning [74]	46.8	67.9	52.2
KD-BCE [79]	55.8	54.6	55.5
ILT [104]	57.7	56.8	57.5
SI [80]	59.9	58.8	59.6
JTAD-SD	66.1	65.9	66.0
JTAD	68.8	69.2	**68.9**
TSMU (ours)	67.2	66.9	**67.1**

Performance, after model update, on the old tasks, new task, and the Weighted Average (WA) = $(w_1 \cdot \text{mIoU}_{\text{oldtask}} + w_2 \cdot \text{mIoU}_{\text{newtask}})/20$, where $w_1 = 15$ (5 per ecoregion, (ETF+MWCF+NAD)) and $w_2 = 5$ (NWFM).

TABLE IV:
PERFORMANCE COMPARISON (MIOU) FOR MODEL UPDATE METHODS STARTING FROM {MWCF+NAD+NWFM} TO {ETF}

Naive Transfer baseline: Ref mIoU (old) = 67.4 ; Ref mIoU (new) = 47.4			
Methods	[MWCF + NAD + NWFM] (old) → [ETF] (new)		Weighted Average (WA)
Retraining [18]	43.2	69.7	49.9
Feature Extraction [75]	67.4	54.8	65.5
Fine Tuning [74]	48.4	68.1	53.3
KD-BCE [79]	58.6	57.8	58.4
ILT [104]	60.1	60.8	60.3
SI [80]	60.9	61.2	61.0
JTAD-SD	66.9	68.2	67.2
JTAD	69.5	71.8	**70.1**
TSMU (ours)	68.6	69.1	**68.7**

WA = $(w_1 \cdot \text{mIoU}_{\text{oldtask}} + w_2 \cdot \text{mIoU}_{\text{newtask}})/20$, where $w_1 = 15$ (5 per ecoregion, (MWCF+NAD+NWFM)) and $w_2 = 5$ (ETF).

TABLE V:
PERFORMANCE COMPARISON (MIOU) FOR MODEL UPDATE METHODS STARTING FROM {NAD+NWFM+ETF} TO {MWCF}

Naive Transfer baseline: Ref mIoU (old) = 68.8 ; Ref mIoU (new) = 44.2			
Methods	[NAD + NWFM + ETF] (old) → [MWCF] (new)		Weighted Average (WA)
Retraining [18]	45.8	67.4	51.2
Feature Extraction [75]	68.8	53.6	65.3
Fine Tuning [74]	47.8	68.7	53.0
KD-BCE [79]	57.1	56.6	56.9
ILT [104]	61.7	62.2	61.9
SI [80]	62.9	63.3	63.0
JTAD-SD	66.6	67.1	66.7
JTAD	68.7	69.9	**69.0**
TSMU (ours)	67.6	68.8	**67.9**

WA = $(w_1 \cdot \text{mIoU}_{\text{oldtask}} + w_2 \cdot \text{mIoU}_{\text{newtask}})/20$, where $w_1 = 15$ (5 per ecoregion, (NAD+NWFM+ETF)) and $w_2 = 5$ (MWCF).

As expected, the performance of **retraining** [18] the model on only the new data results in significantly

TABLE VI:
PERFORMANCE COMPARISON (MIOU) FOR MODEL UPDATE METHODS STARTING FROM {NWFM+ETF+MWCF} TO {NAD}

Naive Transfer baseline: Ref mIoU (old) = 69.1 ; Ref mIoU (new) = 21.8			
Methods	[NWFM + ETF + MWCF] (old) → [NAD] (new)		Weighted Average (WA)
Retraining [18]	36.9	66.2	44.3
Feature Extraction [75]	69.1	35.9	62.1
Fine Tuning [74]	40.8	68.8	47.8
KD-BCE [79]	58.5	55.4	57.7
ILT [104]	60.2	63.9	61.2
SI [80]	64.2	64.9	64.4
JTAD-SD	66.2	67.9	66.6
JTAD	71.8	70.1	**71.3**
TSMU (ours)	69.9	67.3	**69.3**

WA = $(w_1 \cdot \text{mIoU}_{\text{oldtask}} + w_2 \cdot \text{mIoU}_{\text{newtask}})/20$, where $w_1 = 15$ (5 per ecoregion, (NWFM+ETF+MWCF)) and $w_2 = 5$ (NAD).

degraded performance on the old data. The highest degradation can be observed in row 1 in Table VI, where the mIoU on the old task after updating the model is 36.9%, dropping by 34%. This is again because NAD undergoes a significant domain shift in the pixel distribution. The next highest degradation due to retraining is observed in row 1 in Table III, where the mIoU on the old task after updating the model is 37.4%, with a degradation of 30%. This is also because NWFM contains landslides that share the distribution with all the other three ecoregions, as shown in Figure 4.

The performance of **Feature Extraction** [75] on the new tasks is much worse than for the **retraining** method. However, there is no performance degradation on the old tasks. This indicates that shared parameters (which are not updated for the new ecoregion in this baseline) will indeed require (careful) updating in order to achieve good performance on old and new tasks. The lowest new task mIoU after update is 35.9% in row 2 in Table VI. This is because NAD is significantly different from other ecoregions. The encoder (which contains the shared parameters) that is trained to extract good features for the old task is unable to extract effective features for capturing landslides in the NAD ecoregion.

Fine Tuning [74] (row 3 in Tables III-VI), shows catastrophic forgetting on old tasks but appears to be uniformly better than the **retraining** approach on both old and new tasks. Bacause the main difference between the **finetuning** and **retraining** baselines is the addition of new task-specific parameters, this shows that this architectural choice can be important (although it continues to show that the shared parameters need to be updated carefully to avoid catastrophic forgetting).

The three parameter isolation methods **KD-BCE** [79], **ILT** [104] and **SI** [104] perform significantly better than retraining, feature extraction and fine tuning on old tasks. This shows the importance of adding the constraint of Knowledge Distillation [54] to the objective function, reducing the effect of catastrophic forgetting. However, the performance on the new task is poor when compared to Fine Tuning. We think the reason is that these methods assume parameter independence and share all parameters (and hence indicates that task-specific parameters are indeed necessary). But the performance is generally better than retraining and feature extraction on new tasks. However, the weighted average of these tasks is higher than the above mentioned methods, with the average over all the four models being 57%, 60%, and 62% for KD-BCE, ILT, and SI, respectively. SI is higher than the others as it uses two additional constraints: PAS and RCS in its objective function.

The upper bound baselines **JTAD-SD** and **JTAD** (which require old data to be revisited, along with the new data) are considered next. JTAD and JTAD-SD differ only on the architecture, as explained in Section IV-D. Both these methods performed better than the other baselines, which is expected considering that they operate in a less restricted setting. The mean weighted average of JTAD-SD across all the tables is $\approx 67\%$, while that of JTAD is $\approx 69\%$. Approaching this level of performance would be the desired behavior for an update method that is restricted from revisiting the old data.

The proposed **TSMU** compares favorably with JTAD and JTAD-SD and often outperforms JTAD-SD, illustrating that a good choice of architecture can overcome JTAD-SD's built-in advantage of having all the data at once.

Additionally, it can be observed that for TSMU there is actually a boost in the performance of old tasks with increases of 0.6%, 1.2%, and 0.8% in tables III, IV, and VI (compared to the reference mIoUs in these tables) but a decrease of 1.2% in Table V. This appears to show that the updates to the shared parameters (performed when new data are received) may be beneficial to the old tasks. However, this difference becomes much more clear when we compare JTAD (trained on all data) to the reference mIoU in each table (trained on three ecoregions). For example, in Table III, the performance of JTAD on ETF, MWCF, and NAD is 68.8% and it outperforms the reference mIoU (66.6%) that is trained on EFT+MWCF+NAD at the same time. The only difference between them is the addition of NWFM into the training data of JTAD, which shows that adding this ecoregion into the training data improved performance on the other ecoregions. Similarly, JTAD vs. reference mIoU are 69.5% vs. 67.5% in Table IV, 68.7% vs. 68.8% in Table V, and 71.8% vs. 69.1% in Table VI **further shows that adding ecoregions outside a region of interest can improve performance on the region of interest**.

The design of TSMU leverages the following observations from other model update methods: (1) Shared parameters set is sensitive to changes in input data distribution as shown by feature extraction and fine tuning, so, we need a controlled constraint to train the shared parameters to avoid catastrophic forgetting. (2) Increasing data diversity by including data from new ecoregions results in increased accuracy on older ecoregions, as shown by increased accuracy of Fine Tuning when compared to complete retraining from scratch. (3) Knowledge Distillation provides a constraint mechanism to control the change in model parameters, thus decreasing catastrophic forgetting. This is shown by KD-BCE, ILT and SI. (4) task specific parameters are necessary for increased

performance in scenarios where sequential flow of data undergoes significant domain shift as shown by Fine Tuning and Feature Extraction. TSMU uses task specific parameters, feature extraction in incremental freeze training step and fine tuning with knowledge distillation in the joint fine tuning step.

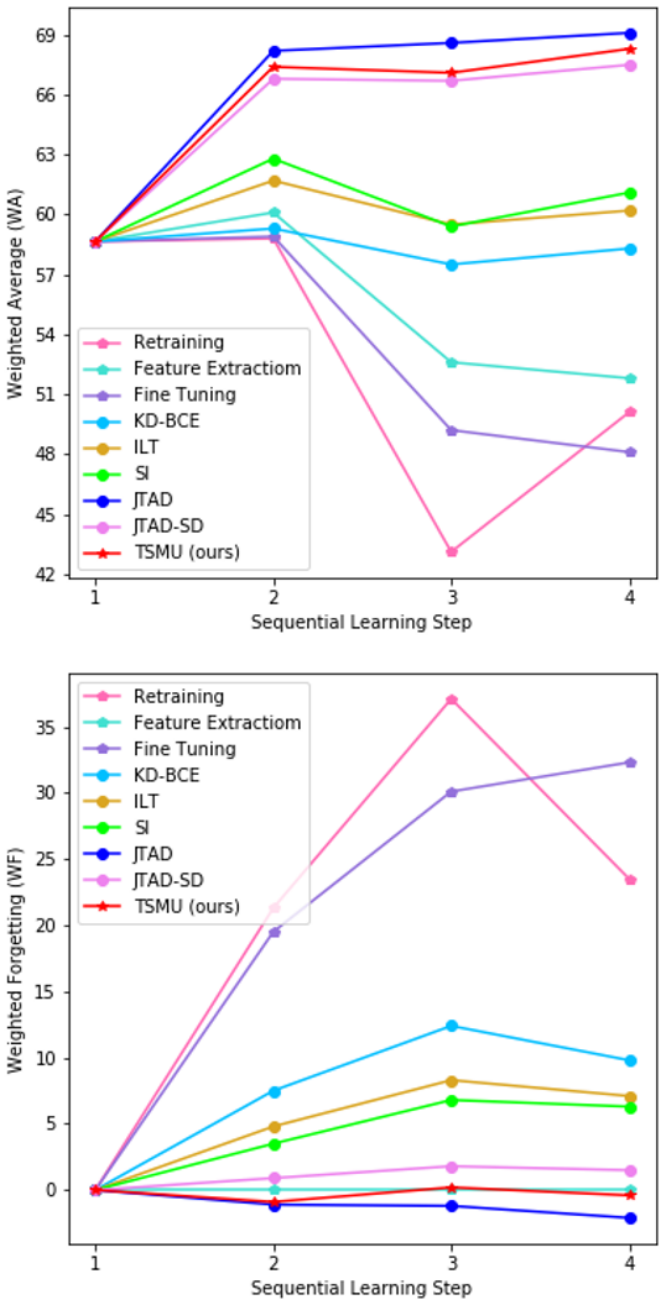

Fig. 10: **Performance Comparison Plots for Sequential Model Update methods.** The two plots show the **Weighted Average (WA) (higher is better)** and **Weighted Forgetting (WF) (lower is better)** metrics over four sequential learning steps from ETF to MWCF to NAD to NWFM.

2) Sequential: As discussed in Methods, the sequential data collection setting (IV-C) tests the ability of the models to perform multiple updates, as would be expected in a deployed setting. The base architecture (Section III-B2) is initially trained on ETF. This is followed by the model being updated on data from MWCF, followed by NAD and finally NWFM. Task specific decoders (Figure 7) are appended for each update as shown in Figure 5. Higher values of Weighted Average (WA) mIoU and lower values of Weighted Forgetting (WF) denote reduction of catastrophic forgetting and increased performance of continual learning.

The results are shown in Figure 10. Generally, TSMU is the best performing method that does not revisit old data and compares favorably with JTAD and JTAD-SD (both of which do revisit old data as new ecoregions arrive in this sequential setting). The retraining baseline shows the worst performance with a final weighted average (WA) of 50.1%. Feature extraction has no effect on the initial Reference mIoU on ETF as the shared parameter set is not changed. It can be observed that the values of WA decrease over sequential steps, with a final value of 51.8%. This decline can also be observed in Figure 10. Similar to feature extraction, fine tuning turns out be a non-performing strategy in the long term, presenting a final value of only 48.1%. fine tuning shows the compounding effect of catastrophic forgetting over sequential learning steps. Parameter isolation methods KD-BCE, ILT and SI perform better than the previous methods, with the final WA of 58.3%, 60.2% and 61.1%. The trend of WA is increasing over the sequential learning steps for JTAD with the final WA of 67.5%. Similar to LOO experiments, TSMU performs the best on sequential updates as well, with a final WA of 68.3%. This is supported by the regularization effect its Joint Fine Tuning step which increases the mIoU of the old task in each update.

For these methods, it can be observed that the value of WA decreases with addition of NAD (step #3), and increases with the addition of NWFM. For all methods except JTAD, it can be observed that the average forgetting rate increases when NAD is added, as also shown in Figure 10. This is because NAD is the most characteristically different ecoregion among the four, as shown in Figure 3. TSMU also has a dip in performance when data from NAD are incorporated, where the WA drops to 67.1% from 67.4%. Addition of NWFM boosts the performance. This is because the data distribution of NWFM overlaps with that of the other three ecoregions as shown in Figure 4.

F. Case Studies

In this section, we visualize landslide segmentation prediction maps obtained as outputs from TSMU. At test time, we use an Autoencoder gate mentioned in section III-C to route the test image through the network to the corresponding ecoregion-specific decoder. The predictions for sample test input images from all ecoregions can be observed, which are output by a trained TSMU network and can be observed in Figure 12. Prediction maps from JTAD are also provided for reference, which can be used as baseline to evaluate the performance of TSMU. It can be seen that the outputs of TSMU closely match the ground truth maps and perform better than JTAD in most of the samples, showing TSMU's effective performance on unseen data points from all ecoregions.

We have already observed quantitative results in tables II to VII, where adding data from a new ecoregion boosts

TABLE VII:
PERFORMANCE COMPARISON (MIOU) FOR SEQUENTIAL MODEL UPDATES STARTING FROM ETF TO MWCF TO NAD TO NWFM

Methods	Naive Transfer baseline: Ref $mIoU_{ETF}$ (old) = 67.7 Ref $mIoU_{MWCF}$ (new) = 49.6			Naive Transfer baseline: Ref $mIoU_{ETF}$ (old) = 67.7 Ref $mIoU_{MWCF}$ (old) = 49.6 Ref $mIoU_{NAD}$ (new) = 28.4				Naive Transfer baseline: Ref $mIoU_{ETF}$ (old) = 67.7; Ref $mIoU_{MWCF}$ (old) = 49.6 Ref $mIoU_{NAD}$ (old) = 28.4; Ref $mIoU_{NWFM}$ (new) = 39.2				
	ETF (old) → MWCF (new)		WA_1	ETF (old) → MWCF (old) → NAD (new)			WA_2	ETF (old) → MWCF (old)→ NAD (old)→ NWFM (new)				WA_3
Retraining [18]	46.4	67.2	56.8	30.6	32.8	65.9	43.1	44.2	46.6	41.4	68.2	50.1
Feature Extraction [75]	67.7	52.4	60.1	67.7	52.4	37.8	52.6	67.7	52.4	37.8	49.1	51.8
Fine Tuning [74]	48.2	69.6	58.9	37.6	42.7	67.1	49.2	35.4	40.8	45.8	70.1	48.1
KD-BCE [79]	60.2	58.3	59.3	55.3	56.2	60.8	57.5	57.9	58.8	54.2	62.3	58.3
ILT [104]	62.9	60.4	61.7	59.4	58.3	60.9	59.5	60.6	59.9	55.2	64.8	60.2
SI [80]	64.2	61.3	62.8	60.9	60.1	57.2	59.4	61.4	60.8	56.9	65.2	61.1
JTAD-SD	66.1	67.5	66.8	65.9	66.9	67.8	66.7	66.2	67.5	66.5	69.8	67.5
JTAD	68.8	67.6	**68.2**	68.9	69.1	67.8	**68.6**	69.8	70.1	67.2	69.2	**69.1**
TSMU (ours)	68.6	66.2	**67.4**	67.2	67.1	66.9	**67.1**	68.1	68.9	65.1	70.8	**68.3**

A model with our base architecture (Section III-B2) is initially trained on ETF until convergence. Ref $mIoU_{ETF}$ is the mIoU of this model when evaluated on the test set of ETF. The same trained models is then evaluated on test sets of MWCF, NAD and NWFM, which give the corresponding Ref $mIoU_{MWCF}$, Ref $mIoU_{NAD}$ and Ref $mIoU_{NWFM}$. These are used as baselines to analyse the performance of different model update methods when data from MWCF, NAD and NWFM are incorporated sequentially to update this base model.

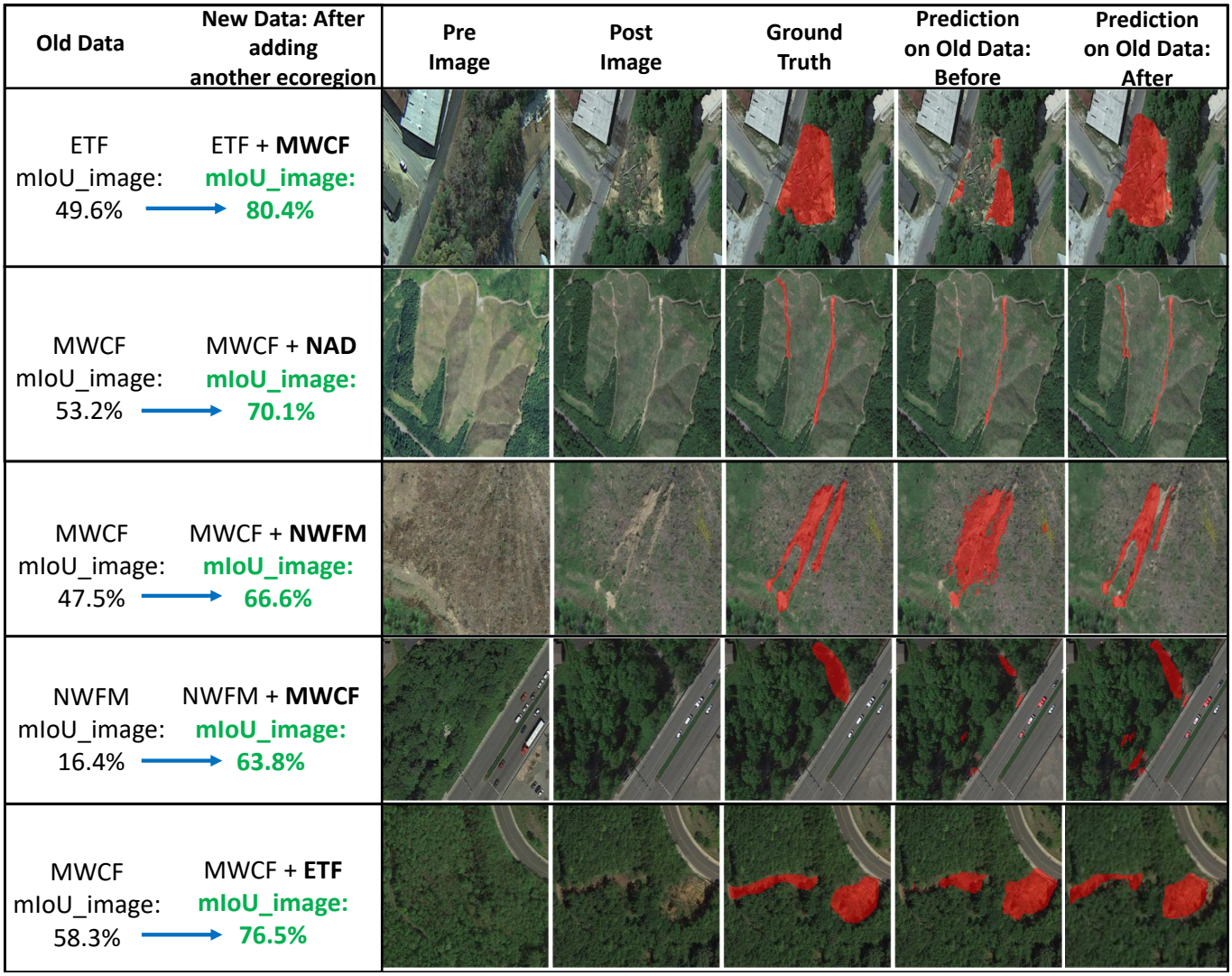

Fig. 11: **Increase in Spatial diversity of data increases model performance on a given ecoregion.** From left to right, pre-event image, post-event image, human-annotated landslide pixel-wise map overlayed on post image (ground truth), prediction of model trained on old ecoregion on an image from old data, prediction of same model on the same image when data from new ecoregion is sequentially added. The figure shows sample predictions of the base architecture of TSMU before and after addition of data from a new ecoregion. It can be observed that sequential addition of data from new ecoregions boosts the performance of the model on already trained ecoregions. This image is best seen in color. The excel file containing the coordinates of the landslides used in this figure will be available in our GitHub link for data.

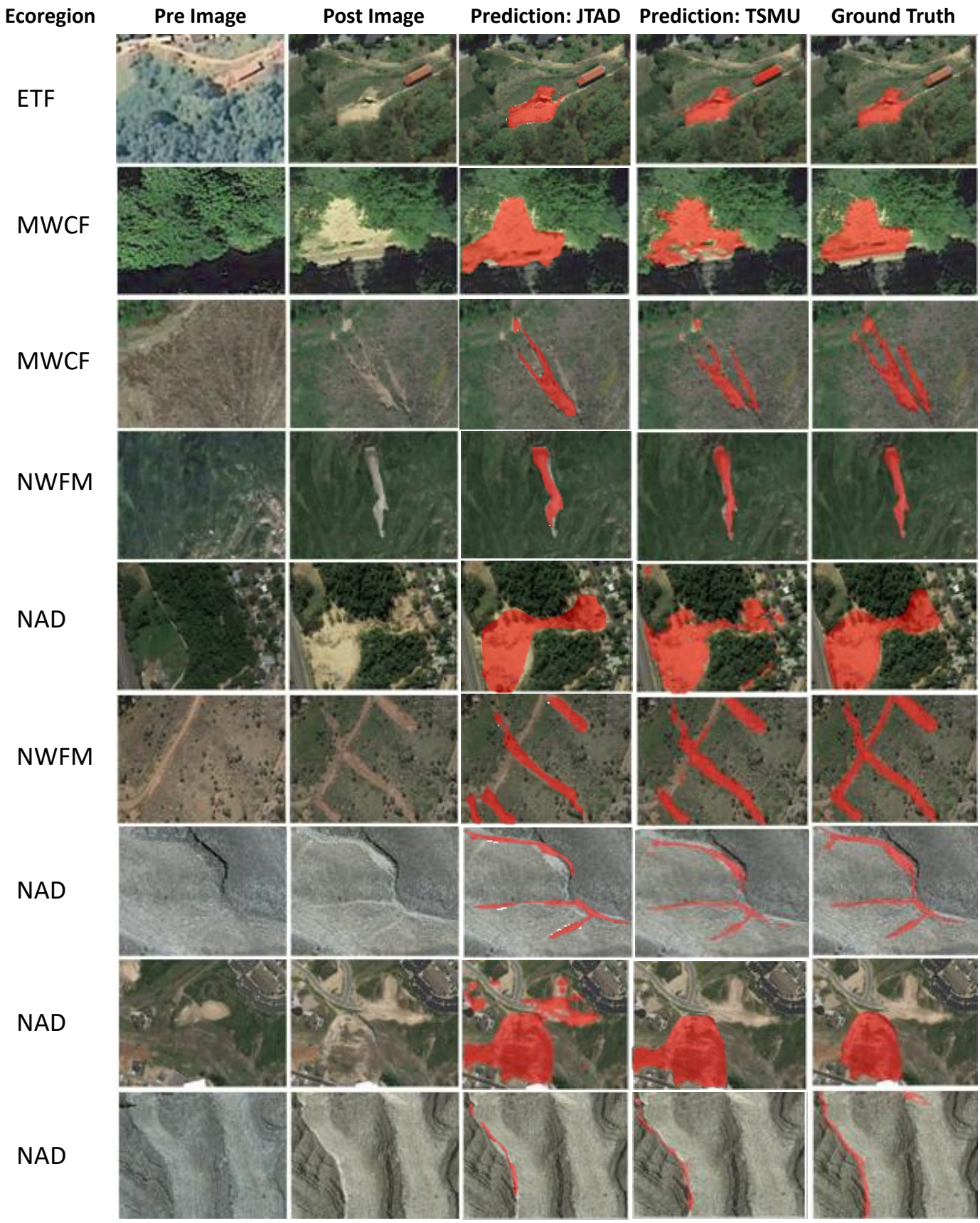

Fig. 12: **Comparative visualization of prediction map samples of JTAD and TSMU from all four ecoregions.** From left to right, pre-event image, post-event image, human-annotated landslide pixel-wise map overlayed on post image(ground truth), prediction of JTAD model trained on all data and prediction of the corresponding task-specific decoder of TSMU after being trained on all data. This image is best seen in color. The excel file containing the coordinates of the landslides used in this figure will be available in our GitHub link for data.

performance on the the older ecoregions. We notice a regularization effect during the joint fine tuning step of the TSMU algorithm (Algorithm 1), which boosts the performance on older ecoregions. This is qualitatively shown in Figure 11. A model is initially trained on one of the ecoregions and then introduced to a new ecoregion. We notice an increase in mIoU of some images belonging to older ecoregions when the model is sequentially updated for the new data.

We notice that data from the ecoregion NAD is characteristically dissimilar (Figure 4) to the data from the other three ecoregions. The landslides in NAD are much thinner, have very low contrast from the background and have lower salience (percentage of coverage of landslide pixels in a given image (Table I)). We train a model on one of the ecoregions other than NAD. We then sequentially add the data from NAD and continually train our model using TSMU algorithm. We observe that this model performs significantly better on narrower landslides from the other three ecoregions (Figure 11). We observe similar behavior when data from all other ecoregions are incorporated sequentially. The improved regions in the prediction maps can be seen clearly. The corresponding increase in the mIoUs are shown as well. ETF and MWCF help boost performance in images that have shadows and forests. NAD helps improve performance on thinner landslides. NWFM boosts the overall performance on all ecoregions due to its diversity as NWFM's distribution overlaps with the other three ecoregions as shown in Figure 4.

V. Conclusion

We asked "**does a model trained on a small homogeneous ecoregion generalize to other ecoregions for which it has not been trained?**". Our Naive Transfer results clearly showed that a model trained on a small region would almost always fail, often drastically, on datasets collected from characteristically different regions. This problem also exists for other geoscientific object identification as well as modeling [105], [106] tasks.

The answer to the second question, "**does increasing the spatial coverage and diversity of data improve model performance in a given ecoregion?**", is **Very Likely**, but the correct methods are required to harness such benefits. It seems a naive method (retraining, feature extraction or fine tuning) leads to significant forgetting so such an improvement would be nonexistent. However, the proposed TSMU (and especially JTAD) can benefit from the diversity in data. If we start from small data (one ecoregion, as shown in Figure 10), the benefit of including more data was more steady and prominent. The synergy between data from different regions is due to the nature of deep learning and the fact that there is underlying commonality between the appearances of landslides even in vastly different environments. This synergy implies, if one has only one ecoregion of interest, one could still collect data both inside and outside that region to improve model performance on that region. Conversely speaking, the best performance of a model cannot be achieved by training on a small, homogeneous region.

With respect to the third question **"what is an efficient and highly-performant way to incorporate new batches of geospatial training data from different environments to update a semantic segmentation model?"**, we proposed an algorithm TSMU that favorably retains memory of the old task during learning sequential batches of data, resulting in a well-generalized, yet highly efficient (in terms of memory and training time), algorithm. This algorithm is not only relevant to landslides but also to other geoscientific tasks. Geoscientific and especially remote sensing datasets can extraordinarily large and the proposed algorithm can also serve as a means to reduce memory footprint so that the problem is tractable with the current graphical processing units.

The TSMU algorithm has considerable value for practical applications in landslide hazards. In particular, this new DL approach can efficiently identify previously unrecognized areas of historical landsliding from existing satellite imagery, and with limited training it can be used to rapidly assess areas of potential landsliding following a widespread triggering event (e.g., major storm or earthquake).

All the supplemental information including code and data - excel files containing coordinates of all (1) landslide events used in our experiments, (2) landslide events used in our figures and (3) landslide events in the USGS database are made available in our GitHub link - https://github.com/deepLDB/landslide-detection.

Acknowledgments

This research was conducted with the help of a Google AI Impact Challenge award. The authors would like to thank other team members who contributed to data collection, including Ningdong Zhang, Jiangtao Liu, Guanlin He, Dapeng Feng, Sean McNally and Shivansh Rao. We would also like to thank Stephen Slaughter and Rex Baum for insightful comments on earlier versions of this work. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government or Google.org.

REFERENCES

[1] M. Dilley, R. Chen, U. Deichmann, A. Lerner-Lam, M. Arnold, J. Agwe, P. Buys, O. Kjekstad, B. Lyon, and G. Yetman, "Natural disaster hotspots: A global risk analysis," *World Bank Disaster Risk Management Series*, vol. 5, pp. 1–132, 01 2005, http://hdl.handle.net/10986/7376.

[2] M. J. Froude and D. N. Petley, "Global fatal landslide occurrence from 2004 to 2016," *Natural Hazards and Earth System Sciences*, vol. 18, no. 8, pp. 2161–2181, 2018, https://doi.org/10.5194/nhess-18-2161-2018.

[3] I. P. on Climate Change, *Climate Change 2013 – The Physical Science Basis: Working Group I Contribution to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2014, doi:10.1017/CBO9781107415324.

[4] E. M. Fischer and R. Knutti, "Anthropogenic contribution to global occurrenceof heavy-precipitation andhigh-temperature extremes," *Nature Climate Change*, vol. 5, no. 6, p. 560–564, 2015, https://doi.org/10.1038/nclimate2617.

[5] P. Reichenbach, M. Rossi, B. D. Malamud, M. Mihir, and F. Guzzetti, "A review of statistically-based landslide susceptibility models," *Earth-Science Reviews*, vol. 180, pp. 60–91, 2018, https://doi.org/10.1016/j.earscirev.2018.03.001.

[6] N. Prakash, A. Manconi, and S. Loew, "A new strategy to map landslides with a generalized convolutional neural network," *Scientific Reports*, vol. 11, no. 1, pp. 1–15, 2021, https://doi.org/10.1038/s41598-021-89015-8.

[7] B. B. Mirus, E. S. Jones, R. L. Baum, J. W. Godt, S. Slaughter, M. M. Crawford, J. Lancaster, T. Stanley, D. B. Kirschbaum, W. J. Burns, R. G. Schmitt, K. O. Lindsey, and K. M. McCoy, "Landslides across the usa: Occurrence, susceptibility, and data limitations," *Landslides*, vol. 17, no. 10, pp. 2271–2285, Oct 2020. [Online]. Available: https://doi.org/10.1007/s10346-020-01424-4

[8] O. Ghorbanzadeh, T. Blaschke, K. Gholamnia, S. R. Meena, D. Tiede, and J. Aryal, "Evaluation of different machine learning methods and deep-learning convolutional neural networks for landslide detection," *Remote Sensing*, vol. 11, no. 2, p. 196, 2019, https://doi.org/10.3390/rs11020196.

[9] T. Lei, Y. Zhang, Z. Lv, S. Li, S. Liu, and A. K. Nandi, "Landslide inventory mapping from bitemporal images using deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 6, p. 982–986, 2019.

[10] L. Bragagnolo, L. Rezende, R. da Silva, and J. Grzybowski, "Convolutional neural networks applied to semantic segmentation of landslide scars," *CATENA*, vol. 201, p. 105189, 2021.

[11] N. Prakash, A. Manconi, and S. Loew, "Mapping landslides on EO data: Performance of deep learning models vs. traditional machine learning models," *Remote Sensing*, vol. 12, no. 3, p. 346, 2020, https://doi.org/10.3390/rs12030346.

[12] K. Crowston, "Amazon mechanical turk: A research tool for organizations and information systems scholars," in *Shaping the future of ICT research: methods and approaches*. Springer, 2012, pp. 210–221.

[13] T. Pei, S. Nagendra, S. Banagere Manjunatha, G. He, D. Kifer, T. Qiu, and C. Shen, "Utilizing an interactive ai-empowered web portal for landslide labeling for establishing a landslide database in washington state, usa," in *EGU General Assembly Conference Abstracts*, 2021, pp. EGU21–13 974.

[14] K. Stacke, G. Eilertsen, J. Unger, and C. Lundström, "A closer look at domain shift for deep learning in histopathology," *CoRR*, vol. abs/1909.11575, 2019. [Online]. Available: http://arxiv.org/abs/1909.11575

[15] Q. Pham, C. Liu, and S. HOI, "Online continual learning under domain shift," 2021. [Online]. Available: https://openreview.net/forum?id=iTeUSEw5rl2

[16] T. Lesort, "Continual learning: Tackling catastrophic forgetting in deep neural networks with replay processes," *CoRR*, vol. abs/2007.00487, 2020. [Online]. Available: https://arxiv.org/abs/2007.00487

[17] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[19] F. Mantovani, R. Soeters, and C. Van Westen, "Remote sensing techniques for landslide studies and hazard zonation in europe," *Geomorphology*, vol. 15, no. 3-4, pp. 213–225, 1996.

[20] A. Carrara and L. Merenda, "Landslide inventory in northern calabria, southern italy," *Geological Society of America Bulletin*, vol. 87, no. 8, pp. 1153–1162, 1976.

[21] F. Guzzetti, M. Cardinali, P. Reichenbach, and A. Carrara, "Comparing landslide maps: A case study in the upper tiber river basin, central italy," *Environmental management*, vol. 25, no. 3, pp. 247–263, 2000.

[22] F. Guzzetti, A. C. Mondini, M. Cardinali, F. Fiorucci, M. Santangelo, and K.-T. Chang, "Landslide inventory maps: New tools for an old problem," *Earth-Science Reviews*, vol. 112, no. 1, pp. 42–66, 2012, https://doi.org/10.1016/j.earscirev.2012.02.001.

[23] Z. Y. Lv, W. Shi, X. Zhang, and J. A. Benediktsson, "Landslide inventory mapping from bitemporal high-resolution remote sensing images using change detection and multiscale segmentation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 5, pp. 1520–1532, 2018, doi: 10.1109/JSTARS.2018.2803784.

[24] Z. Ma, G. Mei, and F. Piccialli, "Machine learning for landslides prevention: a survey," *Neural Computing and Applications*, Nov 2020. [Online]. Available: https://doi.org/10.1007/s00521-020-05529-8

[25] W. Qi, M. Wei, W. Yang, C. Xu, and C. Ma, "Automatic mapping of landslides by the resu-net," *Remote Sensing*, vol. 12, no. 15, p. 2487, 2020.

[26] A. Mondini, F. Guzzetti, P. Reichenbach, M. Rossi, M. Cardinali, and F. Ardizzone, "Semi-automatic recognition and mapping of rainfall induced shallow landslides using optical satellite images," *Remote Sensing of Environment*, vol. 115, no. 7, pp. 1743–1757, 2011.

[27] W. Yang, P. Shi, and L.-Y. Liu, *Identifying Landslides Using Binary Logistic Regression and Landslide Detection Index*, 10 2013, pp. 781–789, 10.1007/978–3–642–32 238–9_85.

[28] W. Yang, M. Wang, and P. Shi, "Using MODIS NDVI time series to identify geographic patterns of landslides in vegetated regions," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 4, pp. 707–710, 10.1109/LGRS.2012.2 219 576, 2013.

[29] P. Lu, Y. Qin, Z. Li, A. C. Mondini, and N. Casagli, "Landslide mapping from multi-sensor data through improved change detection-based Markov random field," *Remote Sensing of Environment*, vol. 231, p. 111235, 2019, https://doi.org/10.1016/j.rse.2019.111235. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425719302548

[30] A. Stumpf and N. Kerle, "Object-oriented mapping of landslides using random forests," *Remote Sensing of Environment*, vol. 115, no. 10, pp. 2564–2577, 2011.

[31] J. Nichol and M. Wong, "Satellite remote sensing for detailed landslide inventories using change detection and image fusion," *International Journal of Remote Sensing*, vol. 26, no. 9, pp. 1913–1926, 2005, https://doi.org/10.1080/01431160512331314047.

[32] 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 Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.

[33] U. Đurić, M. Marjanović, Z. Radić, and B. Abolmasov, "Machine learning based landslide assessment of the Belgrade Metropolitan Area: Pixel resolution effects and a cross-scaling concept," *Engineering Geology*, vol. 256, pp. 23–38, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0013795218311207

[34] S. Nagendra, S. B. Manjunatha, D. Kifer, T. Pei, W. Li, K. Lawson, H. Nguyen, T. Qiu, S. Tran, and C. Shen, "Constructing a large-scale landslide database across heterogeneous environments using task-specific model updates," vol. 2020, pp. NH030–0010, 2020.

[35] Q. Hu, Y. Zhou, S. Wang, F. Wang, and H. Wang, "Improving the accuracy of landslide detection in "off-site" area by machine learning model portability comparison: A case study of jiuzhaigou earthquake, china," *Remote Sensing*, vol. 11, no. 21, p. 2530, 2019.

[36] S. Tavakkoli Piralilou, H. Shahabi, B. Jarihani, O. Ghorbanzadeh, T. Blaschke, K. Gholamnia, S. R. Meena, and J. Aryal, "Landslide detection using multi-scale image segmentation and different machine learning models in the higher Himalayas," *Remote Sensing*, vol. 11, no. 21, p. 2575, 2019, https://doi.org/10.3390/rs11212575.

[37] C. Ye, Y. Li, P. Cui, L. Liang, S. Pirasteh, J. Marcato, W. N. Gonçalves, and J. Li, "Landslide detection of hyperspectral remote sensing data based on deep learning with constrains," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 12, pp. 5047–5060, 2019.

[38] B. Yu, F. Chen, and C. Xu, "Landslide detection based on contour-based deep learning framework in case of national scale of Nepal in 2015," *Computers & Geosciences*, vol. 135, p. 104388, 2020, https://doi.org/10.1016/j.cageo.2019.104388.

[39] L. Zhu, L. Huang, L. Fan, J. Huang, F. Huang, J. Chen, Z. Zhang, and Y. Wang, "Landslide susceptibility prediction modeling based on remote sensing and a novel deep learning algorithm of a cascade-parallel recurrent neural network," *Sensors*, vol. 20, no. 6, p. 1576, 2020.

[40] S. Nagendra, S. Banagere Manjunatha, C. Shen, D. Kifer, and T. Pei, "An efficient deep learning mechanism for cross-region generalization of landslide events," in *AGU Fall Meeting Abstracts*, vol. 2020, 2020, pp. NH030–0010.

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[42] J. Liu, C. Shen, T. Pei, K. Lawson, D. Kifer, S. Nagendra, and S. B. Manjunatha, "A new rainfall-induced deep learning strategy for landslide susceptibility prediction," in *AGU Fall Meeting 2021*. AGU, 2021.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[44] L. P. Soares, H. C. Dias, and C. H. Grohmann, "Landslide segmentation with u-net: Evaluating different sampling methods and patch sizes," 2020, https://arxiv.org/abs/2007.06672.

[45] T. Lei, Y. Zhang, Z. Lv, S. Li, S. Liu, and A. K. Nandi, "Landslide inventory mapping from bitemporal images using deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 6, pp. 982–986, 2019.

[46] P. Amatya, D. Kirschbaum, T. Stanley, and H. Tanyas, "Landslide mapping using object-based image analysis and open source tools," *Engineering Geology*, vol. 282, p. 106000, 2021.

[47] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[48] K. Fang, D. Kifer, K. Lawson, D. Feng, and C. Shen, "The data synergy effects of time-series deep learning models in hydrology," *arXiv preprint arXiv:2101.01876*, 2021, https://arxiv.org/abs/2101.01876.

[49] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3752–3761.

[50] A. M. N. Taufique, C. S. Jahan, and A. Savakis, "Conda: Continual unsupervised domain adaptation," *arXiv preprint arXiv:2103.11056*, 2021.

[51] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, "Remind your neural network to prevent catastrophic forgetting," in *European Conference on Computer Vision*. Springer, 2020, pp. 466–483.

[52] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[53] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020, https://doi.org/10.1109/CVPR.2019.00584.

[54] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[55] S. Özgün, A.-M. Rickmann, A. Guha Roy, and C. Wachinger, *Importance Driven Continual Learning for Segmentation Across Domains*, 09 2020, 10.1007/978-3-030-59861-7_43, pp. 423–433.

[56] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *arXiv preprint arXiv:1909.08383*, 2019.

[57] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *arXiv preprint arXiv:1705.08690*, 2017.

[58] D. L. Silver and R. E. Mercer, "The Task Rehearsal Method of Life-Long Learning: Overcoming Impoverished Data," in *Canadian Conference on AI*. Springer, 2002, https://doi.org/10.1007/3-540-47922-8_8, pp. 90–101.

[59] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, pp. 3987–3995, 10.5555/3 305 890.3 306 093.

[60] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 144–161, 10.1007/978–3–030–01 219–9_9.

[61] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," *arXiv preprint arXiv:1703.08475*, 2017.

[62] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European Conference on Computer Vision (ECCV)*. IEEE, 2018, pp. 532–547.

[63] J. Xu and Z. Zhu, "Reinforced continual learning," *arXiv preprint arXiv:1805.12369*, 2018.

[64] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

[65] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[66] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7120–7129, 2017.

[67] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *arXiv preprint arXiv:1812.00420*, 2018.

[68] S. Lee, J. Ha, D. Zhang, and G. Kim, "A neural dirichlet process mixture model for task-free continual learning," *arXiv preprint arXiv:2001.00689*, 2020.

[69] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.

[70] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International Conference on Machine Learning*. PMLR, 2018, DBLP:conf/icml/SerraSMK18, pp. 4548–4557.

[71] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–82.

[72] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 647–655. [Online]. Available: https://proceedings.mlr.press/v32/donahue14.html

[73] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2014, https://doi.org/10.1109/CVPRW.2014.131, pp. 806–813.

[74] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2014, pp. 580–587.

[75] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic convnet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1790–1802, 2015.

[76] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *European Conference on Computer Vision*, Springer. IEEE, 2014, pp. 329–344.

[77] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *arXiv preprint arXiv:1411.1792*, 2014.

[78] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[79] O. Tasar, Y. Tarabalka, and P. Alliez, "Incremental learning for semantic segmentation of large-scale remote sensing data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 9, pp. 3524–3537, 2019.

[80] Y. Feng, X. Sun, W. Diao, J. Li, X. Gao, and K. Fu, "Continual learning with structured inheritance for semantic segmentation in aerial imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–17, 2022, 10.1109/TGRS.2021.3076664.

[81] R. C. Sidle and T. A. Bogaard, "Dynamic earth system and ecological controls of rainfall-initiated landslides," *Earth-Science Reviews*, vol. 159, pp. 275–291, 2016.

[82] O. America, "Ecological regions of north america," 1997.

[83] G. McMahon, S. M. Gregonis, S. W. Waltman, J. M. Omernik, T. D. Thorson, J. A. Freeouf, A. H. Rorick, and J. E. Keys, "Developing a spatial framework of common ecological regions for the Conterminous United States," *Environmental Management*, vol. 28, no. 3, pp. 293–316, Apr. 2001. [Online]. Available: https://doi.org/10.1007/s0026702429

[84] T. PEI, S. Nagendra, S. B. Manjunatha, G. He, T. Qiu, D. Kifer, and C. Shen, "Cloud-based interactive database management suite integrated with deep learning-based annotation tool for landslide mapping," in *AGU Fall Meeting 2020*. AGU, 2020.

[85] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.

[86] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds., 2015, https://arxiv.org/abs/1412.6980.

[87] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[88] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[89] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1251–1258.

[90] S. Nagendra, C. Funk, R. T. Collins, and Y. Liu, "Foot Pressure from Video: A Deep Learning Approach to Predict Dynamics from kinematics," *CoRR*, vol. abs/1811.12607, 2018, DBLP:journals/corr/abs-1811-12607. [Online]. Available: http://arxiv.org/abs/1811.12607

[91] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 448–456, 10.5555/3 045 118.3 045 167.

[92] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[93] H. Wu and X. Gu, "Max-pooling dropout for regularization of convolutional neural networks," in *IEEE International Conference on Neural Information Processing*. Springer, 2015, pp. 46–54.

[94] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 10.5555/2 627 435.2 670 313, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[95] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[96] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," in *IEEE Deep Learning and Data Labeling for Medical Applications*. Springer, 2016, pp. 179–187.

[97] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *IEEE Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2017, pp. 240–248.

[98] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015, DBLP:journals/corr/KingmaB14. [Online]. Available: http://arxiv.org/abs/1412.6980

[99] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019, DBLP:conf/iclr/LoshchilovH19. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[100] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[101] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

[102] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *IEEE International Symposium on Visual Computing*. Springer, 2016, pp. 234–244.

[103] A. Vehtari, A. Gelman, and J. Gabry, "Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC," *Statistics and Computing*, vol. 27, no. 5, pp. 1413–1432, 2017.

[104] U. Michieli and P. Zanuttigh, "Incremental learning techniques for semantic segmentation," in *IEEE Proceedings of International Conference on Computer Vision Workshops*. Springer, 2019, pp. 0–0.

[105] K. Ma, D. Feng, K. Lawson, W.-P. Tsai, C. Liang, X. Huang, A. Sharma, and C. Shen, "Transferring hydrologic data across continents – leveraging data-rich regions to improve hydrologic prediction in data-sparse regions," *Water Resources Research*, vol. 57, no. 5, p. e2020WR028600, 2021, https://doi.org/10.1029/2020WR028600, e2020WR028600 2020WR028600. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020WR028600

[106] D. Feng, K. Lawson, and C. Shen, "Mitigating prediction error of deep learning streamflow models in large data-sparse regions with ensemble modeling and soft data," *Geophysical Research Letters*, vol. 48, no. 14, p. e2021GL092999, 2021, https://doi.org/10.1029/2021GL092999, e2021GL092999 2021GL092999. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021GL092999