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

savinay nagendra [1], srikanth banagere manjunatha [2], daniel kifer [2], te pei [2], weixin li [2], kathryn lawson [2], hien nguyen [2], tong qiu [2], sarah tran [2], and chaopeng shen [2]

[1]The Pennsylvania State University
[2]Affiliation not available

October 30, 2023



**Abstract**

We use the landslide inventory database provided by the United States Geological Survey. USGS maintains a database of landslide reports with approximate locations and times, but no images. This is the most extensive data of its kind. We extract satellite images from Google Earth by using this inventory.

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

Savinay Nagendra[1], Srikanth Banagere Manjunatha[1], Daniel Kifer[1], Te Pei[2], Weixin Li[2], Kathryn Lawson[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
[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 dataset with high location accuracy. Such a dataset currently does not exist. Recent small-scale studies using deep learning (DL) to label potential landslide areas in remotely-sensed images showed potential but were based on very small, homogeneous regions with unproven model transferability. In this paper, we address two major challenges for creating a large-scale landslide dataset from satellite images with DL. We show that a DL model trained on one ecoregion would perform well on that ecoregion, but struggle in other regions because of differing landscape characteristics. Hence, model training must be performed on heterogeneous regions in order to be applicable for global landslide mapping. To avoid high computational cost when new ecoregions are added to the model, while also avoiding catastrophic forgetting (where model accuracy degrades for older data), we propose Task-Specific Model Updates (TSMU), an adaptation of the *Learning without Forgetting* framework (previously used for image classification) for semantic segmentation (pixel labeling). Numerical experiments show that TSMU can efficiently update the model using the newly acquired images without losing performance on other ecoregions. In fact, we noted a performance increase, demonstrating that if one has a region of interest, obtaining data from other regions can boost performance. The proposed scheme sets the basis for semi-autonomously compiling an unprecedentedly-large-scale landslide database.**

***Index Terms*—Landslides; segmentation; deep learning; ecoregions**

Fig. 1: A bi-temporal pair of pre-event (left) and post-event (middle) images along with a human-generated landslide annotation (right) for a landslide event in May 2014 near Collbran, CO.

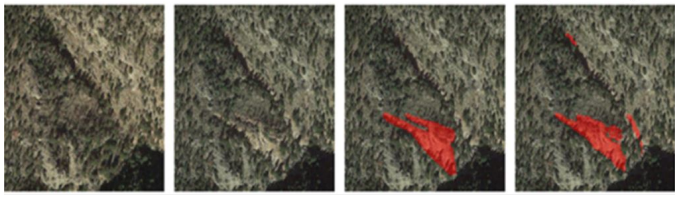

Fig. 2: A more complex example where the human annotator missed some landslide areas (toward the bottom right of the third image) that were detected by a deep learning model (right-most image). The landslide occurred in September 2013 in Colorado Front Range.

## I. Introduction

LANDSLIDES are one of the most widespread 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 impacts of landslides require evaluating landslide susceptibility. Planning and prediction capabilities can be greatly enhanced by large-scale databases of satellite images of landslide events, with accurate location information. In the US, the United States Geological Survey (USGS) maintains a database of landslide reports with approximate locations and times, but no images. This is the most extensive dataset of its kind, but it relies on intensive manual effort to map landslides through field investigations or aerial imagery and is very much incomplete. Furthermore, its spatial location accuracy still leaves much to be desired for planning purposes: recorded coordinates are often not precise enough for determining the landslide setting or for overlaying other datasets to analyze patterns. While each entry can be verified manually from high-resolution satellite images, this process requires significant effort and prevents scaling up. These issues present both an opportunity and a challenge for scalability and coverage, as recently summarized

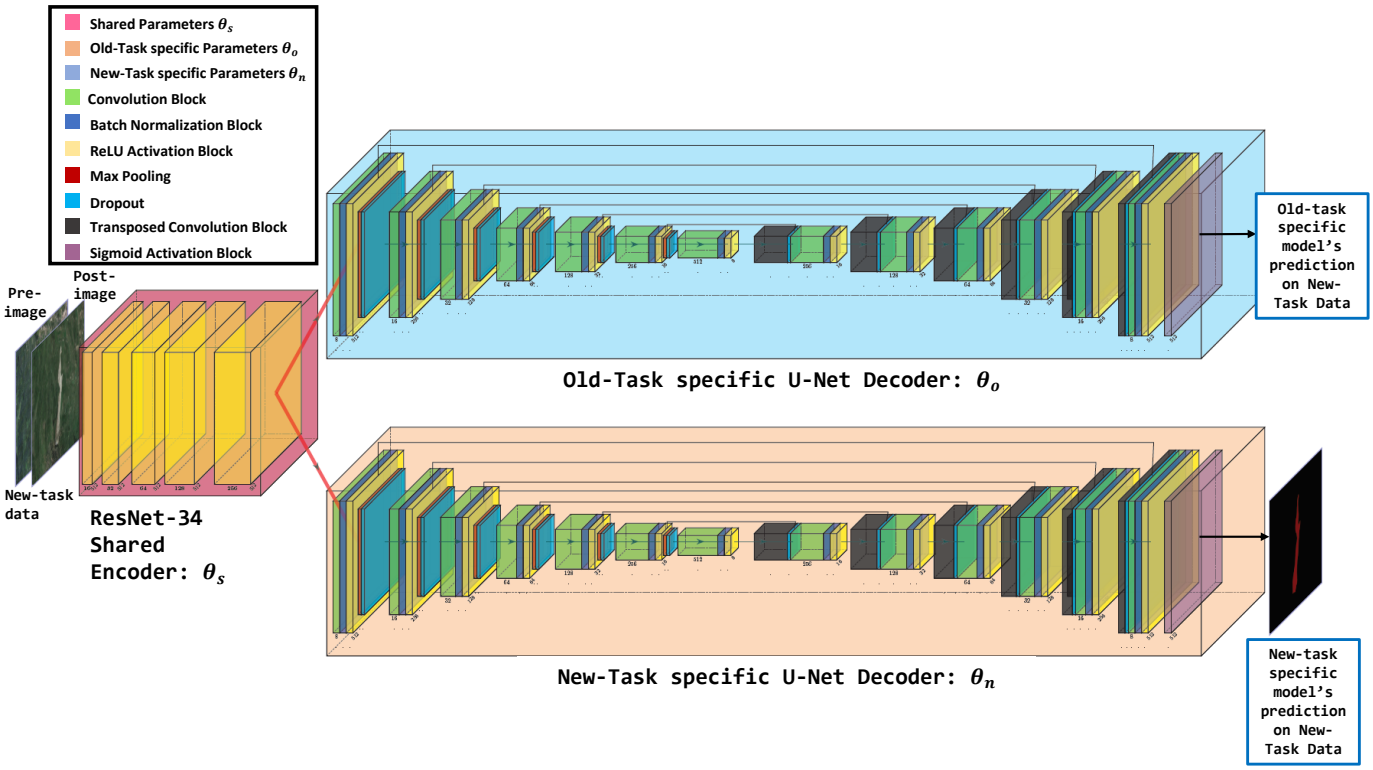


Fig. 3: Illustration of our proposed *Task-Specific Model Updates* (TSMU) algorithm, based on the Learning without Forgetting framework [5]. The semantic segmentation (i.e., pixel labeling) network contains an encoder that extracts features from images and also contains multiple decoders, one for each 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 [6] while the decoders use the U-Net architecture [7]. 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.

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" [8].

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 [9], [10] 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, each study focused on a single small region and had limited testing data (for instance, one study only had 3 testing images [10]).

Using the landslide inventory of approximate location and time provided by USGS, we manually identified 2,509 landslide points and collected 496 "bi-temporal" pairs of satellite images from varied high-resolution sources. Each pair consists of a *pre-image* obtained before the landslide event and a *post-image* obtained after the event (Figure 1). The images mainly came from 4 North American ecoregions [11]: Eastern Temperate Forests (ETF), Marine West Coast Forests (MWCF), North American Deserts (NAD) and Northwestern Forested Mountains (NWFM). Human annotators marked the boundaries of identified landslides on duplicates of the post-event images (Figure 1). Landslides are often irregular in shape, are not always easy to discern in an image (Figure 2), and may be confused with human activities such as logging or construction projects. Thus human annotators need explicit training, and trained annotators still require several minutes to locate a landslide in an image, and several more to process the image

Fig. 4: Pre-event (top) and post-event (bottom) images from four ecoregions in our dataset (left to right): Eastern Temerate Forests (ETF), Marine West Coast Forests (MWCF), North American Deserts (NAD) and Northwestern Forested Mountains (NWFM).

once a landslide has been located. 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 is not as widely available as the orthorectified satellite images used here. For these reasons, building a global landslide image database by hand is clearly infeasible. The training required for human labelers also rules out crowd-sourcing platforms like Amazon Mechanical Turk. However, automated and semi-automated approaches for constructing landslide databases appear more promising, as one could start with a manually-labeled dataset, then train landslide segmentation models and use them to label additional satellite images.

In such a scenario, the first question we would ask is whether a model trained on small homogeneous ecoregions [9], [10] can generalize to other ecoregions for which it has not been trained. Ecoregions are characterized by geographically-distinct features such as vegetation and soil type. Landslides are largely identified due to the disturbance of the land surface and vegetation, and thus these features vary among different ecoregions. As can be seen from Figure 4, ecoregions and their landslides are visually very distinct – they differ in color, texture, landslide size, and visual contrast with the background. For this paper, we performed Leave-One-Out experiments where three ecoregions were used for training and one for testing, rotating the selection and repeating the process so that each ecosystem served as the test region. We then examined how model accuracy responded to characteristically different datasets. In a nutshell, we observed that models can struggle on new ecoregions, and hence a more diverse training set is needed.

The next challenge in building a semi-automated global landslide database is how to update the landslide segmentation model as images from new ecoregions are acquired. Since 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 (see Figure 3). One obvious approach – Option 1 – is to re-train the model on the entire dataset, old and new data combined. This is a computationally expensive and energy-intensive solution [12]. Another approach – Option 2 – is to incrementally train the model using only the new data. However, this can result in *catastrophic forgetting* [13], a phenomenon that is marked by deterioration of performance on ecoregions covered by the old data. Another possibility – Option 3 – avoids catastrophic forgetting by training a new separate model for the new ecoregion. This inherently limits performance as any new knowledge gained cannot be used to improve on the old ecoregions.

To combat these problem, we propose *Task-Specific Model Updates* (TSMU), an adaption of the Learning without Forgetting [5] method (previously used for image classification) for semantic segmentation (pixel labeling). TSMU takes advantage of our model architecture to achieve the two goals laid out in the Learning without Forgetting framework: (1) for efficiency, only new data should be used for training, (2) updates of shared parameters should not cause large changes to predictions on old data. We compare this model with alternatives in terms of accuracy and computational cost. While TSMU is not as computationally efficient as simple incremental training (Option 2), it is much more accurate than Options 2 and 3 and does not exhibit catastrophic forgetting (in fact, it improves accuracy on the old ecoregions). The accuracy compares favorably with retraining on the entire data (Option 1), but TSMU requires significantly less computation.

## II. Related Work

Detection and classification of landslides is critical for hazard analysis [14]. Temporal and spatial occurrences of landslides can be perceived better with comprehensive landslide mapping, which will aid landslide hazard and risk management [15], [16]. 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 as the perspective offered by a distant view provides us a good understanding of the size and extent of a landslide event [17]. Commonly-used approaches for mapping landslides from remote sensing images are briefly summarized below.

### A. Classical Approach

Because landslides are low-likelihood events in space, visual interpretation of landslides from remote sensing images without knowing the accurate location is a labor-intensive empirical technique that requires experience, training, and a systematic methodology [17], [18]. While conventional computer vision and image processing techniques have increased landslide mapping efficiency from visual-interpretation-based methods [17], these semi-automatic methods still require human intervention for each scene and cannot produce landslide maps in an end-to-end fashion. Moreover, the accuracy of these methods can be easily affected by noise and outliers in the image [19]. Two approaches are commonly used in the literature [20]: (1) pixel-based methods, which use spectral information to detect pixels from remote sensing images that correspond to landslides (e.g., [21]–[24]); and (2) object-based image analysis, which uses both spectral and spatial information for landslide identification (e.g., [18], [25]). These two approaches are commonly integrated with image change detection to identify landslides from multi-temporal remote sensing images. [26] used the change detection technique to differentiate landslides from bare rock, soil, and other spectrally-similar features. Machine Learning (ML) techniques such as Logistic Regression, Support Vector Machines, and Random Forests have also been used to improve performance for both pixel-based and object-based image analysis [21], [27], [28]. However, despite the various techniques used to identify landslide pixels from remote sensing images, both pixel-based and object-based approaches require extensive parametric tuning and post-processing, limiting their application at large scales or across different regions.

### B. Deep Learning Approaches and Limitations of Homogeneous Training Data

Deep learning, a branch of ML, has also been used for landslide mapping [20], [29]–[34]. Deep learning techniques are more efficient in terms of automatic feature engineering directly from satellite imagery. [29] used landslide images from multiple seasons from LandSat satellite and DEM data to extract landslides using three ML models. [32] used Sentinel-2 images, DEM data from Lidar, and a derived NDVI layer to map landslides using a modified U-Net model [7]. Due to the sparse availability of high spatial resolution DEMs for automatic landslide detection, especially in the case of some isolated mountainous regions with complex topography, some researchers use only spectral bands of satellite images to identify regional landslides with acceptable accuracy. [20] proposed a deep network called ResU-Net for landslide identification. The ResU-Net model uses residual blocks [6] in the U-Net architecture. The authors claim that the residual learning used in the encoding path of the U-Net model can address data sparsity problems [20]. Given the success of the U-Net architecture in semantic segmentation, it has been used by multiple researchers for landslide mapping. [35] used U-Net to automatically segment landslides in the city of Nova Friburgo. [36] compared three deep learning models – Fully Convolutional Network, Fully-connected Deep Neural Network, and U-Net – to detect landslide areas. [32] 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.

While the abovementioned studies employed various deep networks to detect landslides, each of them focused on a small, homogeneous region, despite the fact that most practical ML applications have favored big data from diverse situations. These small scales might have been inherently limited by the scopes of these studies, but regardless, two uncertainties arise: (i) it is uncertain whether the models trained in these regions are applicable outside of the training region as no assessment took place across regions; (ii) it is unclear if these models achieve optimal performance when they are only trained using data from the region of interest. As deep networks can model highly complex concepts, when training data are from a homogeneous distribution, the model may be overfitted to the peculiarities in this region. 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 [37]. No study thus far has examined the impacts of including images from different regions on vision model performance.

### C. Arrival of Training Data in Temporal Batches

In traditional supervised learning, there is an assumption that all the training data is available together for all tasks. But, in many practical applications, training data arrive in temporal batches. Each new batch might be coming from a different source, making its distribution distinct. For example, we obtain our landslide images in batches corresponding to different ecoregions. There is significant visual diversity across ecoregions due to the differences in physical properties between the geographical locations to which they belong. We use the term *task* to define landslide mapping in one ecoregion. A deep learning model trained on one task often suffers from catastrophic forgetting [13] 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. Overcoming this problem is the focus of the *continual learning* field [38], [39].

Continual Learning approaches can be classified into three categories [40], [41]:

*1) Parameter isolation-based methods:* These methods assign different parameters in a network to each task. This is achieved by dynamic extension of the network [5], [42] or by a fixed architecture [43]. Dynamic architectures can be increasingly memory intensive with every newly added task.

*2) Replay-based methods:* These methods span from naive rehearsal [44] algorithms, which store old data to pseudo-rehearsal methods where generative models are used to approximate previous samples [45].

*3) Regularization-based methods:* Data-focused [5], [46] regularization approaches use data distillation to utilize the knowledge of old tasks to enhance the performance of the model. Prior-focused [13], [47], [48] 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 [48].

Learning without forgetting (LwF) [5] is a multi-task learning training strategy that is a combination of data-focused regularization and dynamic parameter isolation. Proposed for classification problems, it 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 [49] in the training objective function. Our proposed TSMU is an extension of LwF from classification to the more complex problem of semantic segmentation (labelling all pixels in an image). To get the concept behind LwF to work in this setting, we needed to change the network architecture to be better suited to this task. Then, instead of making only new copies of the final layer for each new task, we make copies of the decoder part of our architecture. This usage, in a much more complex problem and leveraging a more powerful network for task-specific predictions, is a good test of the robustness of the principles underlying LwF (and hence TSMU).

### D. Competing Methods

There are other methods in the literature which leverage previously-learned shared parameters to learn new task-

specific parameters [5] - Feature Extraction, Fine-tuning, and Joint Training. These methods are evaluated for comparative analysis with TSMU:

*1) Feature Extraction:* Feature Extraction [50], [51] is a method where outputs of the lower levels of a trained network (such as the convolutional layers) are used as features for a given image, since convolutional layers are believed to extract important visual information from images. These features are then fed into a different (task-specific) network.

Feature extraction can be further improved by *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 [52] 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 [52], [53] and has better performance than retraining the task-specific network from randomly initialized weights [54], [55]. 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 [56], the old, new, and shared parameters are jointly optimized by utilizing samples from every task. Joint training improves generalization on each task by utilizing the domain information in the training data. It is the most computationally-expensive alternative, and can be considered to be the upper bound for TSMU and other continual learning approaches.

## III. Study Area

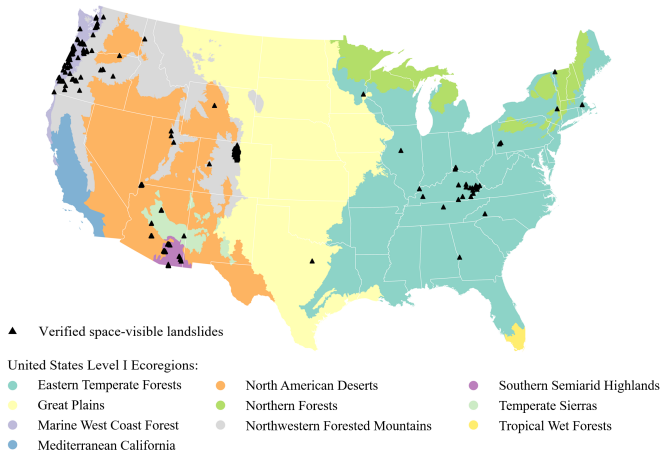


Fig. 5: Distribution of identified space-visible landslides in the present study and Level I ecoregion map for the United States.

Landslides are commonly-seen natural processes that are widely distributed across the North American continent, which has a rich and diverse landscape ranging from forest to desert. In this study, we focused on space-visible landslides that occurred in the contiguous United States and were reported in the USGS landslide inventory [8]. 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) [57], [58]. Thus, the distribution of landslides varies from region to region, favoring different combinations of triggering mechanisms and landscape factors. Although landslides do occur in all fifty United States, we focus here on the conterminous states (i.e., lower 48 states) where landslides are often concentrated along the West Coast, the Appalachian Mountains, and the Intermountain West. An ecoregion defines areas that share similar ecosystems, such as hillslope environment and climate [11]. 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). We focus on landslides that have left a space-visible scarp; hence, landslides such as rockfalls and topples are excluded. Figure 5 shows the distribution of Level I ecoregions and space-visible landslides identified in the present study.

## IV. Dataset Characteristics

Prior studies on landslide segmentation have used datasets with at least one of the following characteristics: (1) they contained very 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 images from geographically diverse regions. We obtained landslide locations using the landslide inventory from the United States Geological Survey (USGS) [8].

This inventory provides approximate locations and times of landslide events. To get post-event images, our data collection team examined these locations to find space-visible landslides with sub-meter resolution satellite images. We filtered out images that had heavy cloud cover or low resolution. For every post-event image found, we obtained a pre-event image. Multiple pre-event images at different times were examined and the clearest pre-event image was chosen. The images were then manually annotated to mark the landslide areas, resulting in 496 pairs of georeferenced pre-event/post-event images with an average size of 1200 by 800 pixels.

Satellite images of rainfall-triggered landslide events in the United States are relatively infrequent (i.e., sparse), and about two-thirds of the landslides in the USGS inventory were in the three West Coast states [8]. Thus, the data had a class imbalance in terms of number of satellite images in each ecoregion: we obtained 27 images for ETF, 30 images for MWCF, 39 images for NAD, and 400 images for NWFM

Histograms of landslide and background pixel intensities for each of the four ecoregions (Figure 6) indicate that each ecoregion, as well as landslides in each ecoregion, had distinct characteristics. Quantitative statistics (Table I) indicate that the distributions were dissimilar, suggesting that a model trained on a subset of ecoregions may not transfer well to ecoregions on which it has not been trained. These analyses also suggest that without care, training a model on new ecoregions could cause its performance to degrade on old ecoregions due to these dissimilarities between ecoregions.

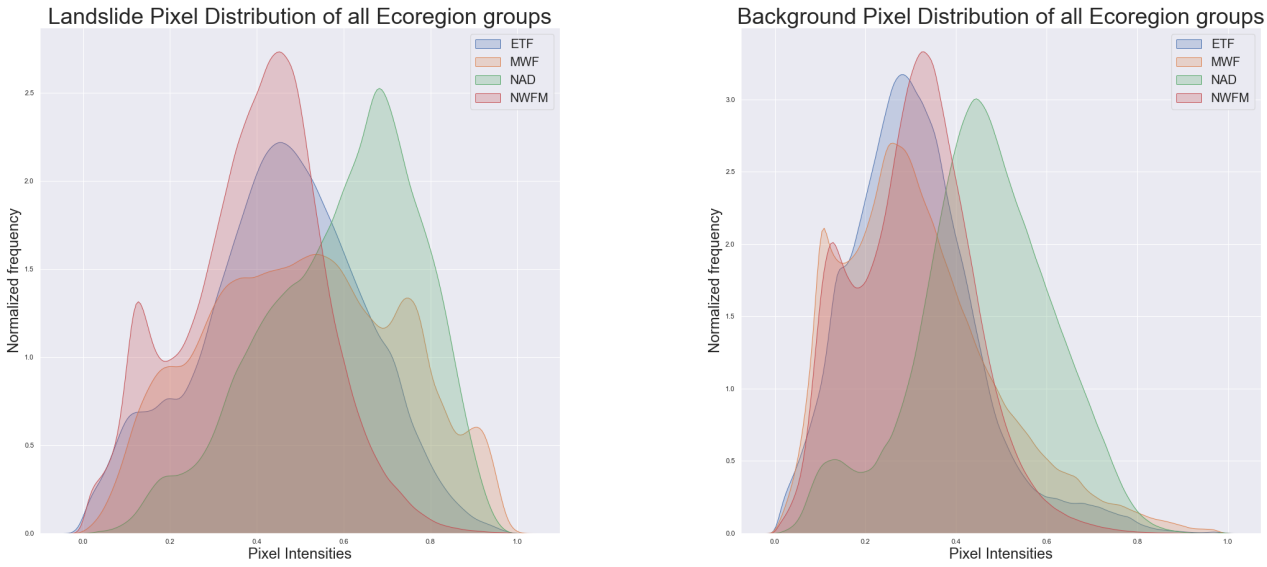


Fig. 6: Pixel-wise Landslide and Background Distributions for ecoregions: Eastern Temperate Forests (ETF), Marine West Coast Forests (MWCF), North American Deserts (NAD) and Northwestern Forested Mountains (NWFM)

TABLE I: Pixel Distribution Statistics

| **Ecoregion** | **# of training image pairs** | **Landslide Pixel Mean** | **Landslide Pixel Standard Deviation** | **Landslide Pixel Median** | **Background Pixel Mean** | **Background Pixel Standard Deviation** | **Background Pixel Median** |
|---|---|---|---|---|---|---|---|
| Eastern Temperate Forests (ETF) | 27 | 0.454 | 0.186 | 0.461 | 0.306 | 0.142 | 0.295 |
| Marine West Coast Forests (MWCF) | 30 | 0.504 | 0.219 | 0.503 | 0.317 | 0.171 | 0.293 |
| North American Deserts (NAD) | 39 | 0.597 | 0.176 | 0.624 | 0.459 | 0.151 | 0.461 |
| Northwestern Forested Mountains (NWFM) | 400 | 0.389 | 0.161 | 0.404 | 0.304 | 0.129 | 0.308 |

The pixel labeling problem was formulated as a binary classification problem for each pixel ("is" vs "is not" a landslide pixel). We refer to the ground truth as a *binary mask*. A data annotation tool called LabelMe [59] was used to manually label the landslide areas in the satellite images. Data was 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).

As a pre-processing step, satellite images were resized to $512 \times 512$ 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.

## V. Methods

### A. Data Processing

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 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). Random augmentations (vertical or horizontal flips of the images) were performed on the remaining images, resulting in the training set.

The validation set was used to tune hyperparameters such as the learning rate and momentum parameter $(\beta)$ of our optimizer (Adam [60]). Validation images were never used to report accuracy metrics. We constructed the validation set from the training data by applying 2 or more of the following random augmentations to each image pair: diagonal flip, horizontal flip, vertical flip, zoom, translation, Gaussian blur. We would like to 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 had slightly different characteristics from the training set, thus reducing the chance of the model overfitting to the training set.

Overall, after augmentation, we obtained approximately 2880 image pairs.

Due to the sparse availability of rainfall-triggered landslides in the United States, there was a class imbalance in terms of the number of satellite images in each ecoregion. In other words, we obtained 400 total image pairs in the Northwestern Forested Mountains (NWFM) ecoregion, but only $\approx 30$ image pairs in each of the other three ecoregions (see Table I). Training a machine learning model with such a skewed data distribution tends to bias a model towards focusing on cases with more data, at the expense of the others. To mitigate this problem, we used a method called "uniform sub-sampling" during training. Specifically, during each training epoch, we randomly sub-sampled 35 image pairs from the NWFM training data. This way, training images for each training epoch were balanced among the 4 ecoregions, but since model training consists of multiple epochs, all of the training images were eventually used.

### B. Metrics and Evaluation Criteria

The model takes as input a pair of pre- and post-event images, and for each pixel outputs a prediction value of 0 or 1 (1 = predicted landslide and 0 = predicted non-landslide).

We employed standard metrics commonly used to evaluate the quality of pixel-labeling tasks: Intersection over Union (IoU), Recall, Precision, and F1 score. Specifically, IoU is the number of correctly predicted landslide pixels (intersection between the prediction and ground truth) divided by the total number of ground truth and predicted landslide pixels (union). Precision is the fraction of predicted landslide pixels that were correct. Recall is the fraction of the true landslide pixels that were correctly captured by the model, and F1 is the harmonic mean of precision and recall. These metrics, mathematically defined in Equations 1-4, were calculated for each image, using the number of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) pixels.

$$\text{Intersection over Union (IoU)} = \frac{\text{TP}}{\text{TP + FP + FN}} \tag{1}$$

$$\text{Precision (P)} = \frac{\text{TP}}{\text{TP + FP}} \tag{2}$$

$$\text{Recall (R)} = \frac{\text{TP}}{\text{TP + FN}} \tag{3}$$

$$\text{F1 Score} = \frac{\text{2*P*R}}{\text{P + R}} \tag{4}$$

For catastrophic forgetting experiments, training consisted of two phases. In the initial phase, the model was trained on a subset of ecoregions. In the second phase, the model was updated with data from the remaining ecoregion(s). Since the competing approaches differed in how the second phase was performed, we evaluated their training costs using the following metrics:

- **Number of epochs**: number of training epochs needed for the model to converge during the second training phase.
- **Time to convergence**: Amount of (wall clock) time taken to converge during the second training phase.
- **Number of image pairs used for training**: The total number of image pairs used in the second phase of training.
- **Number of trainable parameters**: The total number of trainable parameters in the second phase of training.

### C. Model Transferability

Our first set of experiments was designed to gauge whether models trained over homogeneous regions were likely to perform well on new regions. We called these Leave-One-Out experiments because we trained a model using the training data for 3 ecoregions and tested it on the testing data for the remaining ecoregion. Since there were 4 ecoregions in our dataset, this resulted in 4 sets of experiments.

*1) **Base Architecture**:* The base architecture we used for these experiments is illustrated in Figure 3 (without the new-task portion). It consists of an encoder, which transforms a pre- and post-event image pair into an array of features, followed by a decoder which uses the features to label each pixel in the post-event image (1 for landslide pixel and 0 for non-landslide). The input dimensions are $512 \times 512 \times 6$ (each image in a pair is $512x512$ and has 3 RGB channels).

The encoder follows a ResNet-34 [6] architecture. We let $\theta_s$ denote the parameters of this encoder. The convolutional layers [61] of this encoder start with a depth of eight 3x3 convolutional filters and end at 128 filters. Thus the output of the encoder has dimensions $512 \times 512 \times 128$.

The decoder (blue block of Figure 3) is a variant of the U-Net architecture [7]. It consists of a contracting path (first half of the blue block of Figure 3) followed by an expansive path (second half of the blue block). The contracting path is used by the network for further 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 perform feature engineering common to all ecoregions; each new ecoregion will then be given a separate U-Net decoder which will perform futher region-specific feature engineering 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 $3 \times 3$ separable convolution (to reduce the number of trainable parameters) [62] and a Batch Normalization layer [63], followed by a rectified linear

unit (ReLU) [64] activation function. Seven such $R_c$ blocks are used, starting with a depth of 8 filters (channels) and ending with 512 filters. A $2 \times 2$ max pooling [65] 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 [66] 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 $2 \times 2$ upsampling block that uses a transposed convolution (deconvolution) [67] layer. This upsamples the layer's input as well as decreases the number of channels by a factor of two. A skip connection [68] 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 $3 \times 3$ 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 $1 \times 1$ convolution [61] 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 $512 \times 512 \times 1$.

### *D. Continual Learning/Catastrophic Forgetting Experiments*

The next set of experiments considered the situation where a model has been trained on 3 ecoregions and then must be updated with data coming from a 4th ecoregion. We compared our proposed TSMU with other baseline methods described in this section. We started with the baseline methods in order to contrast the different choices that TSMU makes based on the Learning without Forgetting [5] framework. A visual guide to each of the alternatives is shown in Figure 7.

*1) **Baseline Methods**:*

We used six baseline methods. The first four methods do not involve revisiting old data, while the last two do revisit old data.

(i) **Leave One Out (LOO)**: The first option for a baseline, is to simply not update the network in response to the new training data. It is computationally cheap (after the initial training on the old ecoregions) and avoids catastrophic forgetting. The downside is potentially poor accuracy on new ecoregions. This model uses the base architecture described in Section V-C1.

(ii) **Retraining the LOO model (Retraining)**: The next option is to take the model trained on the old ecoregions and to continue training it using just the data from the new ecoregion. In order to guard against drastic forgetting, this subsequent training is done using a small learning rate, which is a common deep learning practice. This model uses the same base architecture as LOO. The purpose of this baseline is to explore performance when no new parameters are added in response to data from the new ecoregion.

(iii) **Fine-tuning**: For this option, a model using the base architecture is trained on the old ecoregions. However, when data for a new ecoregion arrives, a new decoder is added for this ecoregion. Thus the architecture bifurcates, as shown in Figure 7. The old and new ecoregions share the same ResNet encoder. Its parameters are denoted by $\theta_s$. The parameters for the decoder for the old ecoregions are denoted by $\theta_o$ ("$o$" for *old task*) and the parameters for the additional decoder added for the new ecoregion are denoted by $\theta_n$. The model is updated with the new data as follows. The old-task decoder parameters $(\theta_o)$ are frozen (not updated). The shared decoder parameters $\theta_s$ and the new decoder parameters $\theta_n$ are updated (e.g., using gradient descent with the Adam optimizer [60]). Thus after training, the new values $\theta_s'$ of the shared parameters may differ from the old values.

(iv) **Feature Extraction**: This is a variant of the fine-tuning approach where the parameters $\theta_s$ of the shared decoder are not updated. Only the parameters $\theta_n$ of the new decoder are updated. In other words, the shared decoder is viewed as a feature extractor (extracting features from the input). Those features are fed into the new decoder (whose parameters are updated) but the feature extractor remains unchanged.

(v) **Joint Training**: This option extends the feature extraction option. As before, we use the bifurcated structure shown in Figure 7. As in feature extraction, the shared encoder and old ecoregion decoder are trained on the old ecoregions, then the new decoder is added and only its parameters are updated. Afterwards, we combine the new data together with an equally sized random sample of the old data and update *all* of the parameters (this last added step is the difference from feature extraction). The drawback of this approach is that it requires training on old and new data, so can be computationally expensive as the landslide database grows.

(vi) **One model trained on all of the data**: As an alternative to joint training with the bifurcated model, here we train the base architecture. First it is trained on the old ecoregions. When data from a new ecoregion arrives, the model is retrained on *all* of the data (old and new ecoregions). This is also a very computationally-expensive approach and is expected to perform the best. The goal of continual learning is to achieve performance as close to this as possible while significantly reducing the computational expense.

*2) **Task-Specific Model Updates (Proposed Model)**:* We consider two types of architectures. The first is the bifurcated architecture that our other baselines use. This allows for a fair comparison between the training methods for TSMU and the baselines. We also consider an archictecture (Sequential Task-Specific Model Updates in Figure 7) in which there is one shared encoder and a separate decoder for each ecoregion (not just a single decoder for the old ecoregions and one decoder for the new ecoregion). This version is closest to how we would practically deploy TSMU– data for one ecoregion are collected, the model is trained; data for the second ecoregion are collected and the model is updated, and so on.

The underlying training methodology is based on Learning without Forgetting [5] and is outlined in Algorithm 1. The training approach consists of four phases:

A **Initial Training**: As with the baseline methods, train the shared encoder and old-task decoder(s) using gradient descent with the Adam optimizer on the data for the first set of ecoregions. This step sets the initial shared parameters $\theta_s$ and old-task-specific parameters $\theta_o$.

B **Incremental Freeze Training**: This step is the same as for the feature extraction baseline. When data from a

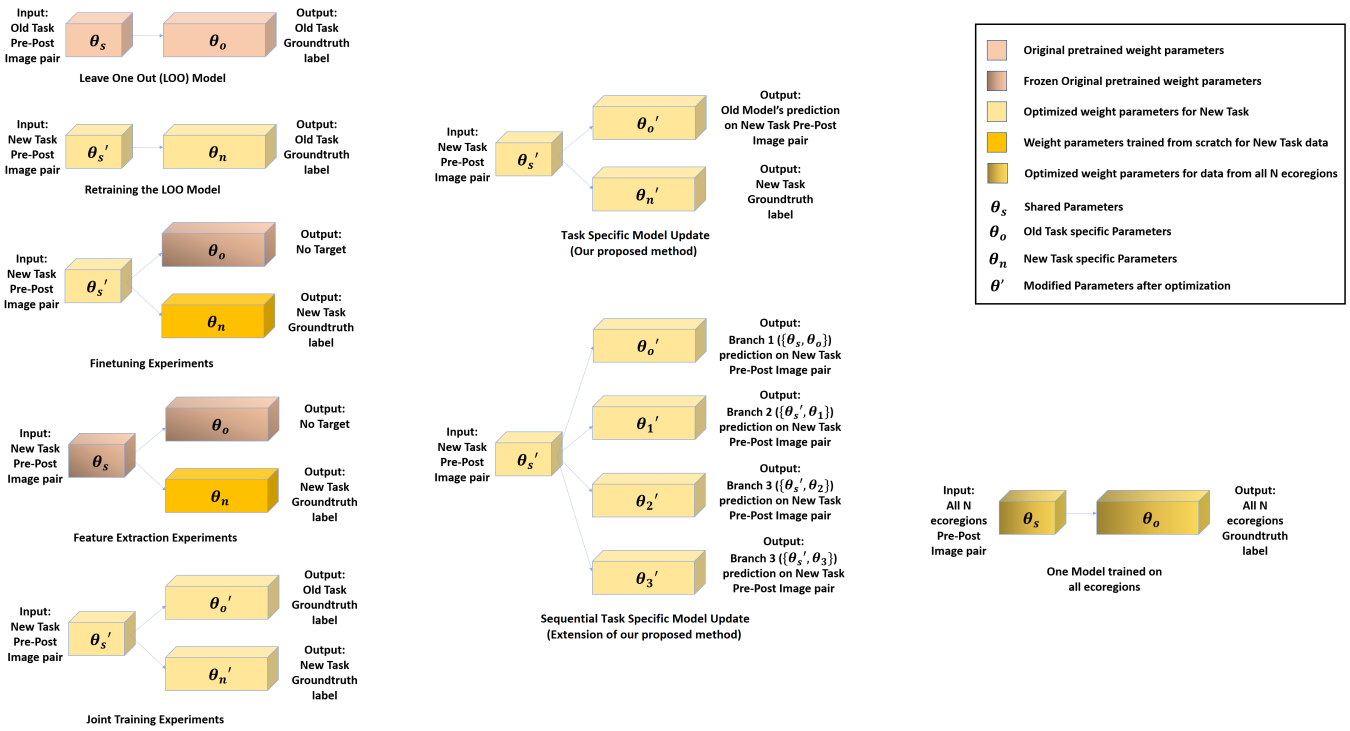


Fig. 7: Overall Block diagram of each of the methods employed. Shared parameters $\theta_s$ correspond to the ResNet-34 feature extractor; $\theta_o$ correspond to the old-task-specific UNet parameters; $\theta_n$ correspond to the new-task-specific UNet parameters. Task-Specific Model Updates and Sequential TSMU are based upon our proposed methods. The model trained on data from all the ecoregions serves as a clear upper bound in our experiments.

new ecoregion is added, we add a new decoder for this ecoregion, with its own parameters $\theta_n$. These parameters are optimized using gradient descent while the shared decoder parameters $\theta_s$ and old-task decoder parameters $\theta_o$ are frozen (unchanged).

C **Stored responses:** For each example (i.e., pre- and post-event image pair) $X_n^i$ in the *new* data $X_n$ we compute the output of the *old-task* decoder(s), and call this $f_o(X_n^i)$; in contrast, the output of the *new-task* decoder is denoted $f_n(X_n^i)$. We store the outputs $f_o(X_n^i)$ for each item $X_n^i$ in the new data. The reason is that we will later train all of the parameters, and ideally, we do not want the old-task predictions to be adversely affected (i.e., on the old-task data, the output of the old-task decoder should be unchanged). However, since we do not want to revisit the old data (to save on computation, since it is larger than the new data), we use a surrogate criterion: the old-task decoder's output $f_o(X_n^i)$ for the new data should not change much during training. Thus we save $f_o(X_n^i)$ for the new data and use it in the next phase. Note that the output layer of our network architecture has a sigmoid activation function, so the output $f_o(X_n^i)$ is a matrix that holds a prediction for each pixel p, and its value is between 0 and 1 (the final landslide/no-landslide prediction is obtained by thresholding – values above 0.5 are converted to 1, and values below are converted to 1).

D **Joint Fine Tuning**: Now we train all of the parameters $\theta_s, \theta_o, \theta_n$ with a small learning rate. The goal is to finetune all of the parameters so that knowledge extracted from the new data can also be applied to the old data. The parameters are trained to minimize a combination of soft Dice loss [69] $L_{dice}$ for improving accuracy on the new data and Knowledge Distillation loss [49] $L_{KD}$ to make sure that the old-task decoder outputs do not stray too far from their previously-stored responses $f_o(X_n^i)$. Let us denote $y_n^i$ to be the ground truth for the i$^{\text{th}}$ pre-/post-even image pair; denote $\hat{y}_n^i$ to be the current output of the new-task decoder; and denote $\hat{y}_o^i$ to be the current output of the old-task decoder for this i$^{\text{th}}$ image pair (in contrast $f_o(X_n^i)$ is the previously stored response for the *initial* weights and does not change during training). The overall loss function that is minimized during training is:

$$\sum_{i=1}^{n} L_{dice}(y_n^i, \hat{y}_n^i) + \lambda \sum_{i} L_{KD}(f_o(X_n^i), \hat{y}_o^i)$$

where the hyperparameter $\lambda$ weights the relative importance of the two loss functions. In our experiments, we simply set $\lambda = 1$. Note that during training, we use weight decay regularization [70] which adds another term to the objective function (see Algorithm 1).

Now, soft Dice loss $L_{dice}$ is a differentiable approximation to the Intersection over Union (IoU) performance measured. It is defined as follows, using the notation that for a given pixel p in data item $X_n^i$, $y_n^i[p]$ is the ground truth label for the pixel and $\hat{y}_n^i[p]$ is the predicted label

**Algorithm 1:** Task-Specific Model Updates training mechanism

**PHASE 1: INITIAL TRAINING**
**Old-Task Training Data:** $\{X_o^i, y_o^i\}$
**Parameters to be trained:** $\{\theta_s, \theta_o\}$
**Parameters frozen:** *None*

Outline:
1. Random Initialization: $\{\theta_s, \theta_o\} \rightarrow \{\theta'_s, \theta'_o\}$
2. Train on old-task data: $\tilde{y}_o^i = f_o(X_o^i|\theta'_s, \theta'_o)$
3. Optimize using Soft Dice Loss: $\theta'_s, \theta'_o \leftarrow \underset{\theta'_s, \theta'_o}{argmin}\{L_{dice}(y_o^i, \tilde{y}_o^i) + R(\theta'_s, \theta'_o)\}$
4. Repeat steps 2 and 3 till saturation.

**PHASE2: INCREMENTAL FREEZE TRAINING**
**New-Task Training Data:** $\{X_n^i, y_n^i\}$
**Parameters to be trained:** $\{\theta_n\}$
**Parameters frozen:** $\{\theta'_s, \theta'_o\} \rightarrow \{\theta'^*_s, \theta'^*_o\}$

Outline:
1. Store response of old-task parameters on **new data**: $f_o(X_n^i|\theta'^*_s, \theta'^*_o) \rightarrow f_o(X_n^i)$
2. Random Initialization: $\{\theta_n\} \rightarrow \{\theta'_n\}$
3. Train on new-task data with frozen $\theta'^*_s$: $\tilde{y}_n^i = f_n(X_n^i|\theta'^*_s, \theta_n)$
4. Optimize **new parameters** on new data using Soft Dice Loss: $\theta'_n \leftarrow \underset{\theta'_n}{argmin}\{L_{dice}(y_n^i, \tilde{y}_n^i) + R(\theta'_n)\}$
5. Repeat steps 3 and 4 till saturation.

**PHASE 3: JOINT FINE-TUNING**
**Training Data (new-task data with stored responses):** $\{X_n, f_o(X_n^i), y_n^i\}$
**Parameters to be trained:** $\{\theta'_s, \theta'_o, \theta'_n\}$
**Parameters frozen:** *None*

Outline:
1. Train on new-task data: $\hat{y}_o^i, \hat{y}_n^i = f(X_n^i|\theta'_s, \theta'_o, \theta'_n)$
2. Optimize **whole model** using Knowledge Distilla--tion and Soft Dice Losses: $\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n \leftarrow \underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{argmin}\{\lambda_o * L_{KD}(f_o(X_n^i), \hat{y}_o^i) + L_{dice}(y_n^i, \hat{y}_n^i) + R(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n)\}$
3. Repeat steps 1 and 2 till saturation.

**Parameters obtained after Training:** $\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n$

for the pixel:

$$I(y_n^i, \hat{y}_n^i) = \sum_{\text{pixels } p} y_n^i[p]\ \hat{y}_n^i[p] \tag{5}$$

$$U(y_n^i, \hat{y}_n^i) = \sum_{\text{pixels } p} (y_n^i[p] + \hat{y}_n^i[p]) \tag{6}$$

$$\text{DiceCoeff}(y_n^i, \hat{y}_n^i) = \frac{2I(y_n^i, \hat{y}_n^i) + \mu_o}{U(y_n^i, \hat{y}_n^i) + \mu_o} \tag{7}$$

$$L_{dice}(y_n^i, \hat{y}_n^i) = 1 - \text{DiceCoeff}(y_n^i, \hat{y}_n^i) \tag{8}$$

The value for the hyperparameter $\mu_0$ was chosen to be 1 using a grid search based on the validation set. $L_{dice}$ is used for the initial training of the old-task data for TSMU as well as the baselines. It is also used during the incremental freeze training portion of TSMU.

The knowledge distillation loss $L_{KD}$ is defined as follows:

$$\hat{y}_o^{i^*}[p] = \frac{\hat{y}_o^i[p]^{1/T}}{\hat{y}_o^i[p]^{1/T} + (1 - \hat{y}_o^i[p])^{1/T}} \tag{9}$$

$$f_o^{i^*}[p] = \frac{f_o(X_n^i)[p]^{1/T}}{f_o(X_n^i)[p]^{1/T} + (1 - f_o(X_n^i)[p])^{1/T}} \tag{10}$$

$$L_{KD}(f_o(X_n^i), \hat{y}_o^i) = -\sum_{\text{pixels } p} f_o^{i^*}[p] \log(\hat{y}_o^{i^*}[p]) \tag{11}$$

The value for the hyperparameter $T$ was chosen to be 2 using a grid search on the validation data.

### *E. Selecting Decoders at Test Time*

At deployment (test) time, a new image pair is typically not labelled with the ecoregion it belongs to. Thus, we have an automated method that determines which ecoregion a new

image pair belongs to (and hence which decoder to use for landslide labeling). We train an *autoencoder* for each region. An autoencoder is a neural network that learns to compress its input (nonlinear dimensionality reduction) and then decompress it (lossy reconstruction). To select the appropriate ecoregion, we pass the image pair through each of the autoencoders. The ecoregion of the autoencoder which has the smallest reconstruction error (measured in terms of mean squared error) is the one that is chosen. We use a 1-hidden-layer fully connected network for the encoder (which compresses the input) and also for the decoder (which reconstructs the input). The decoder has a sigmoid activation in the final layer and is trained using binary cross entropy [71].

### F. Implementation Details

All of our models were trained with an initial learning rate of $1e^{-3}$ for 200 epochs with a batch size of 2 on a NVIDIA RTX 2080 Ti GPU cluster with 16GB memory. The regularization $R$ mentioned in Algorithm 1 is a weight decay of 0.0003 every 20 epochs until the value of $1e^{-9}$ is reached. Data pre-processing is performed before training.

## VI. Results

The previously-described experiments were designed to cover three questions: (1) in order to build a global landslide database, is it sufficient to train a model over a small homogeneous region (will the model transfer to other regions?); (2) can model performance in a region be improved when we add data from more ecoregions?; and (3) how can the model be updated with more data to improve its accuracy while avoiding the computational cost of completely retraining the model with all of the data?

### A. Model Transferability Experiments

The results of the model transferability experiments suggest that these models do not transfer well across ecoregions (Table II). As a reminder, our dataset contained four ecoregions: ETF, MWF, NAD, and NWFM. In the experiments, we trained a model using the training dataset for 3 ecoregions, and then evaluated the model on the test dataset for the 4th ecoregion (which the model hadn't seen before). For example, in the first line of Table II, we trained the model on ETF, MWF, and NAD. For the testing set of these ecoregions (the old-task column), the model achieved an IoU score of 0.638. Such a score indicates fairly good performance (for context, Figure 8 shows some sample pre-event/post-event image pairs along with the ground truth labels and model predictions with IoU scores, from top to bottom, of 0.621, 0.665, 0.632, 0.657 and 0.603). However, when this model was evaluated on the test set of the 4th ecoregion (NWFM), there was a dramatic drop in IoU (0.392). In this instance, the model did not transfer well. We repeated this experiment for each ecoregion, so the 2nd row in Table II shows results when ETF was treated as the new ecoregion, and so on. The largest loss of performance occurred when the North American Deserts (NAD) ecoregion was considered as the new ecoregion (4th row). The other three ecoregions all have forests and therefore are the most homogeneous grouping in these experiments. This underscores the point that training over homogeneous regions is insufficient to building a generic landslide identification model and therefore heterogeneity in the training data should be an important data collection goal.

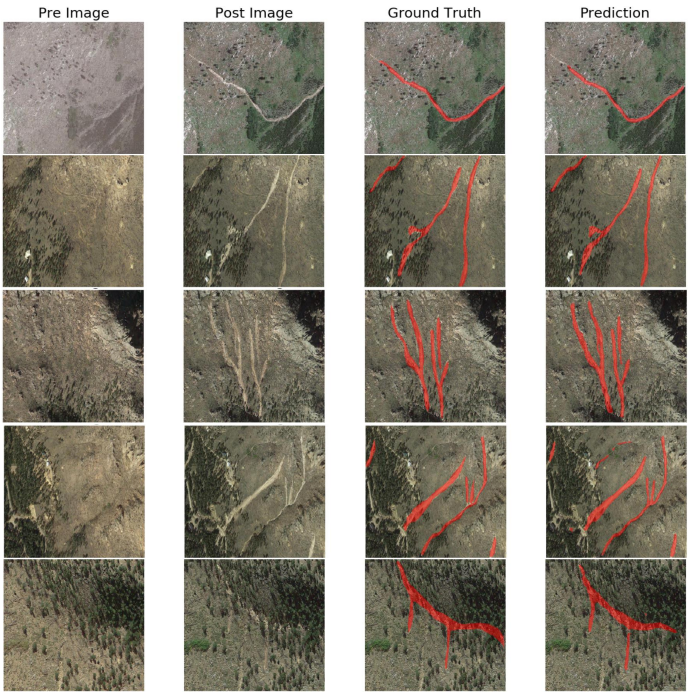


Fig. 8: Sample images containing the pre-event, post-event, ground truth, and prediction labels (left-to-right) from one of the trained models having IoU scores in the range of 0.6. The IoU scores from top to bottom were specifically 0.621, 0.665, 0.632, 0.657 and 0.603.

### B. Model Updating Experiments – Model Quality

When a landslide database is updated with training data from new ecoregions, the model should be updated as well. We have two natural but competing goals: making the model as accurate as possible, and minimizing the amount of computation needed to perform the update. An ideal method would have the best (or nearly the best) accuracy along with significant savings over complete retraining of the model on the entire dataset. We compared TSMU with six baseline methods that have similar architectures but one or more differences in the training approach, thus letting us evaluate the design choices in TSMU. We first report the model quality results and then separately report the runtime costs.

The second baseline model (retraining), which did not add new parameters but simply continued training only on the new data with a low learning rate (thus saving on computations since the more numerous older data were not revisited), worked well for the new region but often exhibited catastrophic forgetting for the old tasks, as demonstrated below (Table III). The first row in Table III shows the case where the data initially contained the ETF, MWF, and NAD ecoregions, and

later on the new ecoregion NWFM was used for retraining. After the model was trained on the first 3 ecoregions, it achieved an IoU of 0.638 (2nd column) on testing data for those ecoregions, while its IoU on NWFM testing data was only 0.392 (third column) because it had not yet seen this ecoregion. After the "retraining" phase was completed, its performance on the new ecoregion improved substantially (increasing from 0.392 to 0.624 in the last column) but exhibited classical catastrophic forgetting: performance on the old ecoregions dropped to 0.332 (second to last column). This behavior was consistent regardless of which ecoregion was treated as the new one, and thus this approach is not viable.

The third baseline model (finetuning) added a new decoder for the new ecoregion and updated the parameters of this decoder, along with the shared encoder, using the new data. Performance was improved for the new task, but moderate forgetting occurred for old tasks (Table IV). After the initial training on 3 ecoregions, the finetuning process was performed to incorporate data from the 4th ecoregion. The last column in Table IV shows that this approach achieved comparable IoU on the new ecoregion to that of the retraining baseline. The second-to-last column in Table IV shows that there was moderate forgetting – performance on the old ecoregions dropped, but not as much as for the retraining baseline.

The fourth baseline model (feature extraction), which added a new decoder for the new ecoregion and updated only the parameters of the decoder and nothing else, showed no forgetting on the old task but suboptimal performance on the new task (Table V). Since the shared encoder was not updated, the old-task decoder made exactly the same predictions on the old ecoregions and so there was absolutely no forgetting (or any other performance change) for the old ecoregions (second-to-last column). However, because the shared encoder was not adapted to extract better features for the new ecoregion, its performance on the new task was slightly worse than for the finetuning baseline.

Having examined the four baseline methods that did not revisit old data, we found that the finetuning approach achieved the best performance on the new data while exhibiting moderate forgetting. Meanwhile, the feature extraction approach effectively guarded against forgetting (at the expense of lower accuracy on the new task). We now compare these to TSMU, which also avoids revisiting the old data.

Results for TSMU show that the algorithm achieved state-of-the-art performance on both old and new tasks, without much compromise (Table VI). Since its training has 3 phases, we show results after each phase. The initial training, which was common to all of the baselines, is shown in the first two columns. The next phase (incremental freeze training) of TSMU updated only the new decoder parameters (hence predictions made by the old-task decoders are unaffected). The middle column shows that this phase improved the performance on the new task (and of course the results on the old ecoregions remained identical to what they were before). The last phase of TSMU used the stored response of the new data on the old-task decoders, and added further improvement in performance for the new ecoregion (last column). Overall, we see this improvement is generally better than all of the 4 baselines that only used the new data. Meanwhile, the second-to-last column shows that the performance on the old tasks also improved.

The results so far have shown that TSMU (which only uses new data for updates) clearly dominated all of the baseline methods that only used new data for updates. The next question is whether it left anything on the table – how much better were computationally-heavy models that revisit all of the data?

The joint training baseline model, which has the same architecture as TSMU including the incremental freeze training part of the model fitting procedure (during this phase, only the newly added parameters were trained), showed a marginal benefit over TSMU. It differs from TSMU in that afterwards, instead of using the stored responses from the new data, it trains all parameters by combining the new data with an equally sized sample of the old data (to reduce computational cost). The results (Table VII) show a slight improvement over TSMU with respect to performance on the new task (last column), as well as a slight improvement on the old task (second-to-last column). We later discuss the computational differences between the two (Table IX) to examine the computational cost of this improvement.

The last baseline that revisited the old data was the single model that was trained on all of the data (in this case there was no distinction between new and old data since they were combined together). Its mean IoU on the combined test set for all ecoregions is shown in Table VIII, along with the IoU values for the other methods over the combined test sets. As expected, this clearly was the best option purely in terms of performance. However, it requires complete retraining every time data for a new ecoregion are obtained, which could be computationally expensive.

The second-to-last row of Table VIII also shows for reference the IoU that was achieved when TSMU was updated sequentially (one ecoregion at a time, instead of training on the first 3 together and then adding the 4th one), referred to as Sequential TSMU. After all of the ecoregions were added, its mean IoU of 0.671 was the second best among all methods.

### C. Model Updating Experiments – Training Time

We now consider the training time needed for these alternative methods (Table IX). For the methods Retraining, Fine-tuning, Feature Extraction, TSMU, and Joint Training, we report the extra time needed to update the model with the 4th ecoregion, while for Sequential TSMU and training with all of the data, we report end-to-end training times.

We see that out of the 4 methods that do not revisit the old data, TSMU is more computationally expensive, adding an extra hour of training time. However, among those methods it is the only one that improved on the new task and old tasks (no forgetting). Joint training revisits some of the old data and achieves slightly higher accuracy metrics than TSMU, however it requires more than twice as much training time. Training on all of the data is by far the most expensive of all.

Moreover, the cost for both joint training and training on all data is both additive (because it only grows as more data

TABLE II: Model Transferability Experiments measuring mean Intersection over Union on a new ecoregion. Test (old task) means the model is tested in the regions where it is trained. Test (new task) means the model is tested in a region withheld from training.

| **Leave-One-Ecoregion-Out: mean Intersection over Union** | | |
|---|---|---|
| **Experiments: Trained on ecoregions** | **test (old task)** | **test (new task)** |
| **ETF + MWF + NAD** | 0.638 | 0.392 (NWFM) |
| **MWF + NAD + NWFM** | 0.643 | 0.498 (ETF) |
| **NAD + NWFM + ETF** | 0.661 | 0.469 (MWF) |
| **NWFM + ETF + MWF** | 0.674 | 0.238 (NAD) |
| **Average** | **0.654** | **0.399** |

are added) and iterative (it needs to happen in every iteration) and thus can grow markedly during a project lifetime.

It is instructive to compare the marginal cost of TSMU adding a new ecoregion (3rd line) with the full cost of adding a region one at a time (e.g., train on ETF, then use the training method to add in MWF, then use the training method to add NAD, etc.) which appears on the second to last line vs. the full cost of training from scratch (last line). The marginal cost of adding a new ecoregion using TSMU is about 3 hours worth of training, so approximately 12 hours for the 4 ecoregions. The marginal cost of adding a new ecoregion when retraining completely on all of the data keeps increasing – if there was a fifth ecoregion to add, then the additional cost would exceed 13.8 hours (last line of the table). Thus the *total* computational cost of adding new ecoregions appears to be linear (i.e., constant marginal cost per ecoregion) while the *total* cost of constantly retraining over all of the data appears to be quadratic (linear marginal cost per ecoregion).

Thus, overall, TSMU appears to strike a good balance between performance and computational expense.

## VII. Discussion and Conclusions

Overall, landslide mapping is a difficult task. It is nontrivial for humans and requires complex modeling (such as deep neural networks). Different ecoregions have different visual characteristics, so mapping potential landslide areas based on satellite data requires a heterogeneous training set in order to be viable on a global scale.

Such a dataset is naturally constructed in pieces, where data from one ecoregion are downloaded and a small number are manually annotated. Subsequently, data from the next ecoregion are downloaded, and the annotation process repeats. Updating a deep learning model as new data are acquired is a non-trivial task. Naive training approaches exhibit catastrophic forgetting, where the network performance degrades for the older data. We evaluated several approaches to defend against this phenomenon and observed that catastrophic forgetting can be avoided without expending the full computational cost on retraining on all of the data collected thus far.

The feature extraction method completely avoids catastrophic forgetting at a reasonable computational cost. However, performance on the old data will not improve as new data are collected. The proposed TSMU method, based on the Learning without Forgetting framework is the computationally cheapest option for which performance improves both on the old and new data. Further improvements are possible, but come at an increased computational cost (at least 2x the training time for the next best alternative).

There are several interesting implications of this finding: (1) not only does the proposed method avoid forgetting, it continues to improve the old-task performance, and (2) all of the information contained in the model update is due to the new data; that is, it is the information in the new data that is helping to improve performance on the old data. The implication of this latter point is that if we have a specific region of interest, we can improve accuracy on this region by obtaining data from other regions even if they have different characteristics. This result also suggests that if we only train on data from one region, as done in the vast majority of papers in the literature, we will not obtain the optimal model for any of these regions. For example, the last row considers the case where the initial ecoregions were the relatively homogeneous forested areas NWFM, ETF, and MWF while the new ecoregion was the North American Desert (NAD). The NAD landslide data helped improve IoU for these old ecoregions from 0.674 to 0.683 even though deserts have significant differences from forested regions. This finding encourages us to shift our mindset of landslide databases from a regional scale to a global scale.

Even though we consider these results promising, some challenges remain. It seems that human supervision is still necessary to some extent, to monitor auto-labeled images for errors and for annotating a few images from new ecoregions. Even skilled geologists may have trouble distinguishing between erosion and landslide features from satellite imagery without stereographic photographs, a high-resolution digital elevation model, or field-site visits. This study leverages widely available satellite imagery to better and more rapidly characterize potential landslides. The approach could present an over-prediction in those specific cases where disturbances from erosional features or construction are not readily distinguishable from landslides. However, this would be a potential limitation for any technique, manual or automated, that relies solely on such data. Additionally, identification of even potential landslide areas that have not been fully vetted by a geologist can be exceedingly useful for many applications (e.g., emergency response to a major storm or earthquake event). Techniques for further reducing the amount of manual effort are needed to rapidly scale such a database. In addition to landslide detection, another challenge is how to use such a database for landslide susceptibility modeling to aid preparation and mitigation efforts.

Our proposed algorithm, TSMU, can efficiently update a model using newly acquired images while improving performance on other ecoregions, demonstrating that if one has a region of interest, obtaining data from other regions can boost performance. This proposed scheme sets the basis for semi-autonomously compiling an unprecedentedly-large-scale landslide database.

TABLE III: Retraining: mean Intersection over Union

| **"Retraining" Baseline: mean Intersection over Union** | | | | | |
|---|---|---|---|---|---|
| **Phase 1 - Initial Training** $\{\theta_s, \theta_o\} \rightarrow \{\theta'_s, \theta'_o\}$ | | | **Phase 2- Retraining** $\{\theta'_s, \theta'_o\} \rightarrow \{\theta'_s, \theta_n\}$ | | **(with low learning rate)** |
| **Experiments: Trained on ecoregions** | **test (old task)** | **test (new task)** | **Experiments: Same model trained only on ecoregion** | **test (old task)** | **test (new task)** |
| **ETF+MWF+NAD** | 0.638 | 0.392 | **NWFM** | **0.332** | **0.624** |
| **MWF+NAD+NWFM** | 0.643 | 0.498 | **ETF** | **0.372** | **0.672** |
| **NAD+NWFM+ETF** | 0.661 | 0.469 | **MWF** | **0.398** | **0.655** |
| **NWFM+ETF+MWF** | 0.674 | 0.238 | **NAD** | **0.401** | **0.641** |
| **Average** | **0.654** | **0.399** | | **0.375** | **0.648** |

TABLE IV: Fine-tuning: mean Intersection over Union

| **Fine-tuning: mean Intersection over Union** | | | | | |
|---|---|---|---|---|---|
| **Phase 1 - Initial Training** $\{\theta_s, \theta_o\} \rightarrow \{\theta'_s, \theta'_o\}$ | | | **Phase 2- Fine-tuning** $\{\theta'_s, \theta'^*_o, \theta_n\} \rightarrow \{\hat{\theta}_s, \theta'^*_o, \theta'_n\}$ $\{\theta'^* =$ ***trained and frozen***$\}$ | | |
| **Experiments: Trained on ecoregions** | **test (old task)** | **test (new task)** | **Experiments: Shared and New task specific parameters trained only on ecoregion** | **test (old task)** | **test (new task)** |
| **ETF+MWF+NAD** | 0.638 | 0.392 | **NWFM** | **0.582** | **0.642** |
| **MWF+NAD+NWFM** | 0.643 | 0.498 | **ETF** | **0.615** | **0.665** |
| **NAD+NWFM+ETF** | 0.661 | 0.469 | **MWF** | **0.604** | **0.658** |
| **NWFM+ETF+MWF** | 0.674 | 0.238 | **NAD** | **0.631** | **0.654** |
| **Average** | **0.654** | **0.399** | | **0.608** | **0.655** |

TABLE V: Feature Extraction: mean Intersection over Union

| **Feature Extraction: mean Intersection over Union** | | | | | |
|---|---|---|---|---|---|
| **Phase 1 - Initial Training** $\{\theta_s, \theta_o\} \rightarrow \{\theta'_s, \theta'_o\}$ | | | **Phase 2- Feature Extraction** $\{\theta'^*_s, \theta'^*_o, \theta_n\} \rightarrow \{\theta'^*_s, \theta'^*_o, \theta'_n\}$ $\{\theta'^* =$ ***trained and frozen***$\}$ | | |
| **Experiments: Trained on ecoregions** | **test (old task)** | **test (new task)** | **Experiments: New task specific parameters trained only on ecoregion** | **test (old task)** | **test (new task)** |
| **ETF+MWF+NAD** | 0.638 | 0.392 | **NWFM** | **0.638** | **0.632** |
| **MWF+NAD+NWFM** | 0.643 | 0.498 | **ETF** | **0.643** | **0.655** |
| **NAD+NWFM+ETF** | 0.661 | 0.469 | **MWF** | **0.661** | **0.647** |
| **NWFM+ETF+MWF** | 0.674 | 0.238 | **NAD** | **0.674** | **0.642** |
| **Average** | **0.654** | **0.399** | | **0.654** | **0.644** |

TABLE VI: Task-Specific Model Updates

| **mean Intersection over Union** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Phase 1 - Initial Training** | | | **Phase 2 - Incremental Freeze Training** | | **Phase 3 - Joint Fine-Tuning** | | |
| **Experiments: Trained on ecoregions** | **original model** $\{\theta_s, \theta_o\} \rightarrow \{\theta'_s, \theta'_o\}$ | | **Experiments: New task-specific parameters trained only on ecoregion (with frozen original model)** | **new model** $\{\theta'^*_s, \theta'^*_o, \theta_n\} \rightarrow \{\theta'^*_s, \theta'^*_o, \theta'_n\}$ $(\theta'^* =$ ***trained and frozen***$)$ | **Experiments: Trained only on ecoregion (all parameters are fine-tuned)** | **new model** $\{\theta'_s, \theta'_o, \theta'_n\} \rightarrow \{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n\}$ | |
| | **test (old task)** | **test (new task)** | | **test (new task)** | | **test (old task)** | **test (new task)** |
| **ETF+MWF+NAD** | 0.638 | 0.392 | **NWFM** | 0.632 | **NWFM** | **0.649** | **0.641** |
| **MWF+NAD+NWFM** | 0.643 | 0.498 | **ETF** | 0.655 | **ETF** | **0.657** | **0.678** |
| **NAD+NWFM+ETF** | 0.661 | 0.469 | **MWF** | 0.647 | **MWF** | **0.672** | **0.668** |
| **NWFM+ETF+MWF** | 0.674 | 0.238 | **NAD** | 0.642 | **NAD** | **0.683** | **0.661** |
| **Average** | **0.654** | **0.399** | | **0.644** | | **0.665** | **0.662** |

TABLE VII: Joint Training: mean Intersection over Union

| **Joint Training: mean Intersection over Union** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Phase 1 - Initial Training** | | | **Phase 2 - Incremental Freeze Training** | | **Phase 3 - Joint Training** | | |
| **Experiments: Trained on ecoregions** | **original model** $\{\theta_s, \theta_o\} \rightarrow \{\theta'_s, \theta'_o\}$ | | **Experiments: New task-specific parameters trained only on ecoregion (with frozen original model)** | **new model** $\{\theta'^*_s, \theta'^*_o, \theta_n\} \rightarrow \{\theta'^*_s, \theta'^*_o, \theta'_n\}$ $(\theta'^* =$ ***trained and frozen***$)$ | **Experiments: Trained on random subset of old data and on new ecoregion (all parameters are fine-tuned)** | **new model** $\{\theta'_s, \theta'_o, \theta'_n\} \rightarrow \{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n\}$ | |
| | **test (old task)** | **test (new task)** | | **test (new task)** | | **test (old task)** | **test (new task)** |
| **ETF+MWF+NAD** | 0.638 | 0.392 | **NWFM** | 0.632 | **NWFM** | **0.653** | **0.645** |
| **MWF+NAD+NWFM** | 0.643 | 0.498 | **ETF** | 0.655 | **ETF** | **0.662** | **0.683** |
| **NAD+NWFM+ETF** | 0.661 | 0.469 | **MWF** | 0.647 | **MWF** | **0.678** | **0.672** |
| **NWFM+ETF+MWF** | 0.674 | 0.238 | **NAD** | 0.642 | **NAD** | **0.684** | **0.668** |
| **Average** | **0.654** | **0.399** | | **0.644** | | **0.669** | **0.667** |

TABLE VIII: Accuracy on all testing data (mean IoU ± std)

| Experiments | Average ± std |
|---|---|
| Leave One Out | 0.591 ± 0.019 |
| Retraining | 0.443 ± 0.015 |
| Fine-tuning | 0.620 ± 0.014 |
| Feature Extraction | 0.648 ± 0.014 |
| Task Specific Model Update | 0.665 ± 0.015 |
| Joint Training | 0.668 ± 0.032 |
| Sequential Task Specific Model Update | 0.671 ± 0.006 |
| Trained with all the data | 0.676 ± 0.015 |

TABLE IX: Additional Training Time

| Experiments | | {ETF+MWF+NAD} + {NWFM} | {MWF+NAD+NWFM} + {ETF} | {NAD+NWFM+ETF} + {MWF} | {NWFM+ETF+MWF} + {NAD} |
|---|---|---|---|---|---|
| **Retraining** | **Number of epochs** | 100 | 100 | 100 | 100 |
| | **Time to convergence (hours)** | ∼1.2 | ∼1.2 | ∼1.2 | ∼1.2 |
| | **Number of images used for training** | 700 | 440 | 500 | 680 |
| | **Number of trainable parameters** | 5,052,945 | 5,052,945 | 5,052,945 | 5,052,945 |
| **Fine-tuning** | **Number of epochs** | 200 | 200 | 200 | 200 |
| | **Time to convergence (hours)** | ∼2.3 | ∼2.3 | ∼2.3 | ∼2.3 |
| | **Number of images used for training** | 700 | 440 | 500 | 680 |
| | **Number of trainable parameters** | 9,796,930 | 9,796,930 | 9,796,930 | 9,796,930 |
| **Feature Extraction** | **Number of epochs** | 200 | 200 | 200 | 200 |
| | **Time to convergence (hours)** | ∼2.3 | ∼2.3 | ∼2.3 | ∼2.3 |
| | **Number of images used for training** | 700 | 440 | 500 | 680 |
| | **Number of trainable parameters** | 9,796,930 | 9,796,930 | 9,796,930 | 9,796,930 |
| **Task Specific Model Update** | **Number of epochs** | 300 | 300 | 300 | 300 |
| | **Time to convergence (hours)** | ∼3.3 | ∼3.3 | ∼3.3 | ∼3.3 |
| | **Number of images used for training** | 700 | 440 | 500 | 680 |
| | **Number of trainable parameters** | 9,796,930 | 9,796,930 | 9,796,930 | 9,796,930 |
| **Joint Training** | **Number of epochs** | 450 | 450 | 450 | 450 |
| | **Time to convergence (hours)** | 7.1 | 6.8 | 6.8 | 7.1 |
| | **Number of images used for training** | 1400 | 800 | 1000 | 1360 |
| | **Number of trainable parameters** | 9,796,930 | 9,796,930 | 9,796,930 | 9,796,930 |
| **Sequential TSMU {Trained on one ecoregion at a time}** | **Number of epochs** | 600 | | | |
| | **Time to convergence (hours)** | 11.6 | | | |
| | **Number of images used for training** | 1820 | | | |
| | **Number of trainable parameters** | 19,278,820 | | | |
| **Trained with all the data** | **Number of epochs** | 400 | | | |
| | **Time to convergence (hours)** | 13.8* | | | |
| | **Number of images used for training** | 2320 | | | |
| | **Number of trainable parameters** | 5,052,945 | | | |

*At least this much time is required for every subsequent ecoregion.

## ACKNOWLEDGMENT

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. 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.

[2] D. Petley, "An analysis of fatal landslides, and the resultant deaths, in 2017," Apr 2018. [Online]. Available: https://blogs.agu.org/landslideblog/2018/04/08/fatal-landslides-2017/

[3] T. Stocker, L. Alexander, and M. Allen, *Climate change 2013: the physical science basis: final draft underlying scientific-technical assessment: Working Group I contribution to the IPCC fifth assessment report*. WMO, IPCC Secretariat, 2013.

[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.

[5] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] 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.

[8] 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

[9] 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.

[10] 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.

[11] 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

[12] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0012825212000128

[18] 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.

[19] 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

[20] 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.

[21] 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.

[22] W. Yang, P. Shi, and L. Liu, "Identifying landslides using binary logistic regression and landslide detection index," in *Earthquake-Induced Landslides*. Springer, 2013, pp. 781–789.

[23] 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, 2012.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] L. P. Soares, H. C. Dias, and C. H. Grohmann, "Landslide segmentation with u-net: Evaluating different sampling methods and patch sizes," 2020.

[36] 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.

[37] 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.

[38] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[39] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent replay for real-time continual learning," *arXiv preprint arXiv:1912.01100*, 2019.

[40] S. Özgün, A.-M. Rickmann, A. G. Roy, and C. Wachinger, "Importance driven continual learning for segmentation across domains," in *International Workshop on Machine Learning in Medical Imaging*. Springer, 2020, pp. 423–433.

[41] 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.

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

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

[44] 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.

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

[46] D. L. Silver and R. E. Mercer, "The task rehearsal method of life-long learning: Overcoming impoverished data," in *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 2002, pp. 90–101.

[47] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3987–3995.

[48] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

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

[50] 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 *International conference on machine learning*. PMLR, 2014, pp. 647–655.

[51] 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*, 2014, pp. 806–813.

[52] 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*, 2014, pp. 580–587.

[53] 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.

[54] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *European conference on computer vision*. Springer, 2014, pp. 329–344.

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

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

[57] 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.

[58] O. AMERICA, "Ecological regions of north america," 1997.

[59] K. Wada, "labelme: Image Polygonal Annotation with Python," https://github.com/wkentaro/labelme, 2016.

[60] 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.

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

[62] 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.

[63] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

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

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

[66] 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, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

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

[68] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," in *Deep learning and data labeling for medical applications*. Springer, 2016, pp. 179–187.

[69] J. Bertels, T. Eelbode, M. Berman, D. Vandermeulen, F. Maes, R. Bisschops, and M. B. Blaschko, "Optimizing the dice score and jaccard index for medical image segmentation: Theory and practice," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, D. Shen, T. Liu, T. M. Peters, L. H. Staib, C. Essert, S. Zhou, P.-T. Yap, and A. Khan, Eds. Cham: Springer International Publishing, 2019.

[70] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.

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