ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TrueBranch: Metric Learning-based Verification of Forest Conservation Projects

Master Thesis

Simona Santamaria

June 26, 2020

Advisors: Prof. Ce Zhang, David Dao.
Department of Computer Science, ETH Zürich

Abstract

International stakeholders increasingly invest in offsetting carbon emissions, for example, via issuing Payments for Ecosystem Services (PES) to forest conservation projects. Issuing trusted payments requires a transparent monitoring, reporting, and verification (MRV) process of the ecosystem services (e.g., carbon stored in forests). The current MRV process, however, is either too expensive (on-ground inspection of forest) or inaccurate (satellite). Recent works propose low-cost and accurate MRV by automatically determining forest carbon from drone imagery, collected by the landowners.

The automation of MRV, however, opens up the possibility that landowners report untruthful drone imagery. TrueBranch, is a metric learning-based verification process. The method is trained to learn a feature space in which images from different locations or times can be easily distinguished despite differing image resolutions. In practice, this feature space can verify the truthfulness of reported drone imagery from forest conservation projects by comparing it with public satellite imagery. This way the truthfulness of drone imagery from forest conservation projects can be verified. We demonstrate empirically that TrueBranch outperforms other feature extraction algorithms by learning a feature space in which 85% of untruthfully reported imagery are identifiable via classification.

Keywords: Deforestation, Climate Change, Image Registration, Image Classification, Convolutional Neural Networks, Representation Learning, Metric Learning, Triplet Loss, Satellite Imagery, Drones, Aerial Imagery.

Acknowledgements

I am very grateful to Professor Prof. Ce Zhang who enabled me to do this thesis at the DS3Lab. I would like to thank David Dao for his support and guidance. Without all his ideas and feedback this work would not have been possible. I am also very grateful to Björn Lütjens, for his critical questions, his pragmatic way of tackling problems and his constructive feedback. Björn and David's enthusiasm and dedication for Climate Change and Machine Learning were very inspiring and motivating.

Finally I would like to thank my housemate and colleague Benjamin Gallusser, for the late night shifts when we tried to debug my code and all our discussions during lunch or coffee break.

Contents

Chapter 1

Introduction

1.1 Motivation

Deforestation and forest degradation has a massive impact on climate change. Agriculture, forestry, and other land use is a key driver, accounting for 23% (12.0 ± 2.9 GtCO2eq yr^{-1}) of total anthropogenic emissions of greenhouse gases during 2007-2016 [21], largely driven by deforestation and forest degradation. Deforestation does not only release carbon (e.g., through slash-and-burn), but also destroys a multitude of other forest ecosystem services: preserving biodiversity, counteracting flooding and soil erosion, filtering water, and offering a livelihood for the local population.

The causes of deforestation are mostly economically driven and can be prevented with financial support for ecosystem services. Reasons for deforestation are: expansion of commercial or subsistence agriculture, logging, fuelwood collection, or livestock grazing [19]). To counteract the economic incentives, payments for ecosystem services (PES) [39] are increasingly [12] provided to forest-conserving or restoring landowners by international stakeholders (e.g., through the governmental UN-REDD program [14] or the commercial voluntary carbon market [12]). To enable payments for ecosystem services a transparent monitoring, reporting, and verification process of the land is necessary.

1.2 Monitoring, Reporting, and Verification of Forest Conservation Projects

Current methods for monitoring, reporting, and verification (MRV) of the landowner-provided forest ecosystem services are either based on 1) on-ground inspection, which is too expensive (about 300 USD/ha [1]), delayed (up to two years), corruptible, and biased [15], 2) satellite, which is low-cost,

but often limited to the binary verification of forest/no-forest cover [17], or 3) drones.

1.2.1 Verification with Drones

Drones have a very high potential as the MRV process is much less expensive than on ground inspection and more accurate then satellite images. A possible verification process is visualized in Figure 1.1.

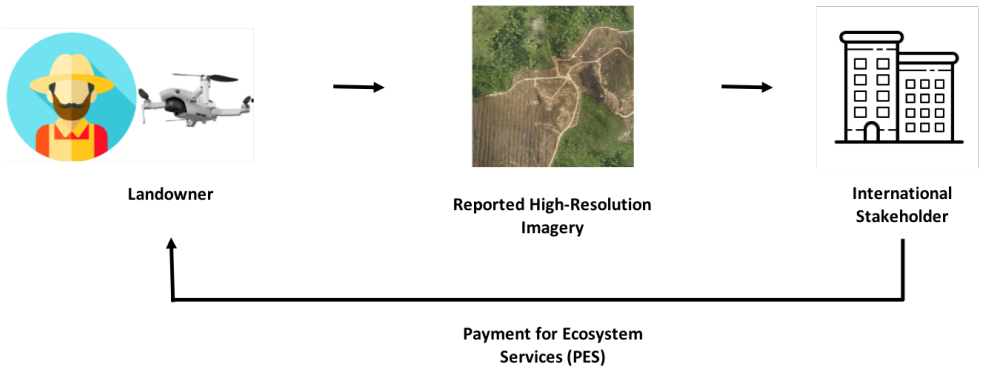


Figure 1.1: Low-cost monitoring via drones - A landowner records his forests with a drone, reports this high resolution images to international stakeholders and they then pay the landowner for his ecosystem services.

Replacing on-ground inspection with remote assessment via drones, however, opens up the possibility of untruthfully reported imagery. Given that the landowner is financially incentivized by PES to report higher forest ecosystem services value (e.g., higher forest cover or biodiversity), the possibility of false reporting exists [38], but is not addressed by previous works.

1.2.2 Attack Vectors

Landowners are financially incentivized to report untruthful drone imagery that displays forest with higher ecosystem services value to receive higher PES. To reason about an algorithm that detects untruthful imagery, we classify common attack vectors with examples, as displayed in Chapter 1.2:

1. Altering drone image **location:** The landowner has land with 50% forest cover and reports imagery from a neighbouring land with 80% forest cover to receive higher valuation.

2. Altering drone image **time stamp:** The landowner reports imagery (e.g., with altered time stamp metadata) from previous flights, before their land has been logged or cleared.

3. Altering drone image **values**: The landowner tricks a neural network-based forest valuation algorithm into estimating higher ecosystem value by altering the image values with sophisticated attacks such a using an image manipulation software (e.g., Adobe Photoshop) or human-imperceptible adversarial perturbations (e.g., PGD [24]).

4. A combination of the above.

Figure 1.2: An overview of possible attack vectors in time, location, and value (left-to-right) that trick an automated forest valuation algorithm to detect high forest cover.

1.3 TrueBranch Verification System

We propose TrueBranch, a metric learning-based verification algorithm that verifies the trustworthiness of reported drone imagery of forest conservation projects. Specifically, TrueBranch aims to verify the truthfulness of drone images via matching them with public satellite images. Matching is proposed to be done in a deeply learned feature space that, ideally, 1) allows for easy distinction of images that display forests of differing ecosystem service value, 2) is robust to adversarial perturbation of the drone image, and 3) generalizes the verification of drone images to other ecosystems (e.g., mangroves or peatlands). Figure 1.3 shows how TrueBranch is embedded in the scheme of automated MRV systems to achieve low-cost, accurate, and trustworthy MRV, which will promote international investments in forest conservation.

1.4 Thesis Contributions

In this section we list the main contributions of this project:

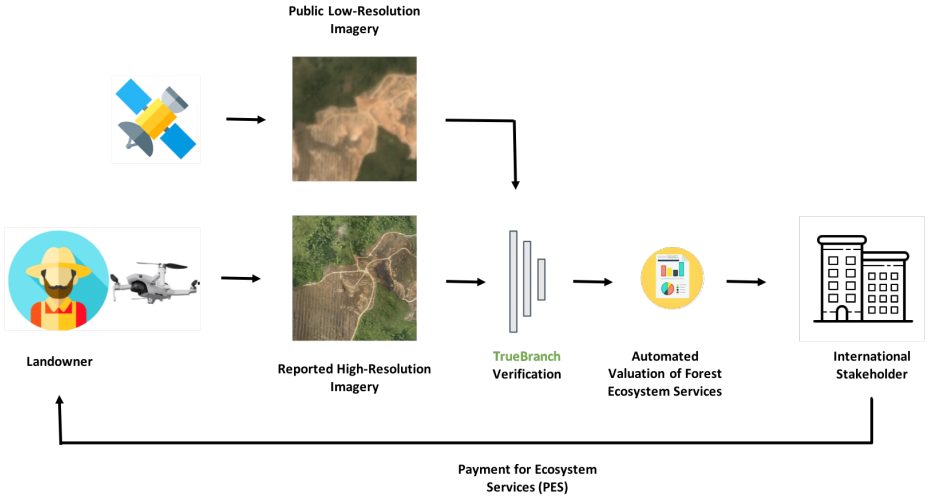


Figure 1.3: An overview of TrueBranch Verification. A landowner takes a high-resolution drone image of their forest and reports the image, time stamp, and location. A metric learning-based algorithm, TrueBranch, verifies the submission with the corresponding low-resolution public satellite image. Another algorithm estimates the forest ecosystem service value based on the verified imagery and international stakeholders provide payments for ecosystem services (PES) to the landowner.

- We propose TrueBranch, a novel way of distinguishing truthfully from untruthfully reported imagery by using low-resolution satellite data to verify the location of high-resolution drone data. The framework for training, validation and testing of deep learning models in Pytorch can be found in the project github repository [4].
- We provide an infrastructure to download satellite and aerial imagery from Google Earth Engine (GEE). The code can be found in the project github repository [4].
- We benchmark a variety of image representations and classification methods for the purpose of identifying untruthfully reported imagery.

1.5 Outline

The work is structured as follows:

- In the first part of chapter two, related work in image registration and geo-localization is presented. The second part of this chapter presents related work in representation and metric learning.

- Chapter three introduces the dataset for this project, including the way the data was collected, the area, the size and the spectral bands that were used.
- In chapter four, several image representations are introduced and possible binary classification methods for TrueBranch Verification presented.
- Chapter five presents the experiments and results obtained with nominal and learned metrics representation, as well as a performance evaluation for different binary classifiers. Furthermore, the classification accuracy of our model is compared to a feature extraction baseline.
- Finally in chapter six conclusions of this project are stated and possible future work proposed.

Chapter 2

Related Work

2.1 Image Registration

Image Registration is the process of aligning one or more images to another image or transforming different sets of data into one coordinate system. There exist many different image registration applications. In this work scene to model registration is relevant, where acquired images are localized in a model. Furthermore, it needs to be distinguished between radar and optical image matching, where for this project only the latter is relevant [6]. The main steps for image registration are: 1. Image pre-processing in order to have the images in the same format, 2. Feature extraction, where depending on the method, specific objects of an image are used as features, 3. Feature matching to find the relationship between the images, 4. Spatial transformation for image restoration and rectification, 5. Image re-sampling to place digital values into new pixel locations.

2.1.1 Feature extraction and Feature matching

The feature extraction itself can be divided in the manual extraction of corresponding control points (CP) and the extraction made by autonomous algorithms where they can be intensity or feature based. Intensity based algorithms compare the intensity patterns in images using correlation metrics while feature based methods find correspondence between image features such as points, lines and contours [16]. Feature detection can be done in many different ways. In the next section the following options are presented:

- Handcrafted Features where features are detected with different algorithms depending on the method.
- Structure extraction with Mathematical Operations using a specific content in the image.

- Deep Learning Approaches where features are learned with neural networks.
- Reinforcement Learning Approaches using an artificial agent.

Handcrafted Features

Both, Tahoun *et al.* [34] and Yang *et al.* [40] investigate the performance of local invariant features for geographic image retrieval and conclude that they are very effective in aerial image retrieval.

Invariant features enable reliable matching despite different transformations. The research community distinguishes between local features where keypoints in a picture are detected and global features that describe the image as a whole, involving all pixels. Local features can be vector or binary based [36]. A listing of possible local feature extractor can be found in Figure 2.1.

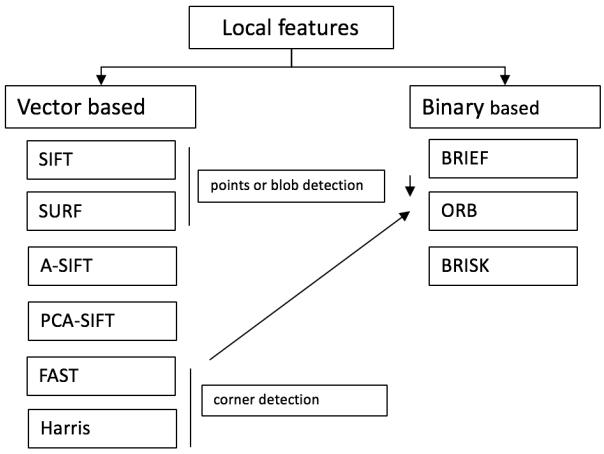


Figure 2.1: An overview of possible local features with ORB as combination of BRIEF and FAST features.

But as the discriminative ability of low-level features is limited, handcrafted features are increasingly replaced by learned features [42]. Nevertheless, due to their low computation power, especially in combination with deep learning, they are still in use. Sharif *et al.* [33] compares CNN features with handcrafted features. Basu *et al.* [7] combines handcrafted features with an unsupervised learning framework (Deep Belief Network).

Structure extraction with Mathematical Operations

Structure extraction makes use of the content in the image where for example roads and rivers are extracted and used as features. These structures

are often detected by parallel edge extraction [32] and morphological operations like Wagner *et al.* [37] uses for individual tree crown delineation. They define the brightest point to be the top of the tree and uses border detection for tree crown detection.

Deep Learning Approaches

Deep learning is characterized by neural networks. Instead of using handcrafted features that are mainly designed on domain-specific knowledge, feature representations are learned from data. Convolutional Neural Networks use interleaving convolutional and pooling layers to extract mid- and high level features from raw images [41].

Deep learning for feature extraction can be done in different ways:

1. Supervised learning with a Convolutional Neural Network in combination with handcrafted features [41].
2. A semi-supervised learning framework where features are first extracted from the image and then fed as input into a model for unsupervised learning. Basu *et al.* [7] for example uses a deep Belief Network that can learn data representations from large amounts of unlabeled data.
3. Using a purely unsupervised learning approach. Three unsupervised feature learning methods used in literature are sparse coding, restricted-Boltzmann machines and spars auto-encoders [13]. Le *et al.* [24] uses locally connected sparse autoencoders to detect objects in a large image dataset, producing state-of-the-art results.

Three state-of-the-art object recognition algorithms used in deep learning are:

- Deep Belief Networks
- Convolutional Neural Networks
- Sparse Autoencoders

2.1.2 Deep Features

A deep feature is generally extracted from the last layer of a CNN, usually produced by a network previously trained on a large computer vision database. According to literature [10][30] they are very powerful as they exploit already trained CNN's to this way not require a large training dataset and producing high level features without having to train a CNN from scratch.

Reinforcement learning Approach

An artificial agent is trained to perform the image registration. This method is only used for very specific complex transformations and thus not many papers are referring to it. Liao *et al.* [25] uses first a greedy supervised algorithm for end-to-end training and then reinforcement learning to align the images by a learned rigid transformation. Krebs at al. uses an artificial agent to optimize the parameters of a deformation model in order to align two images [23].

2.1.3 Spatial Transformation

After the feature extraction and matching, geometric transformations are used to relate the target image to a reference image. Here different transformation models can be used. Detone *et al.* [11] uses a regression HomographyNet to learn the geometric transformation with supervised method. Nguyen *et al.* [28] uses the same approach but with unsupervised learning.

2.1.4 Image Registration for Remote Sensing Data

Remote sensing refers to the use of satellite- or aircraft-based sensor technologies to detect and classify objects on Earth. This data has specific characteristics that make the image registration more challenging than other vision datasets [36][7]:

- The data is often multimodal
- There is a variability in time
- The size of the data
- The data is geolocated
- The data depends on resolution
- The variety in types of sensor data and the conditions during data acquisition
- The lack of a known image models

Dahmane *et al.* [9] researches the transferability of computer vision features to remote sensing applications and presents a convolutional deep approach where deep features are used with a CNN to detect cars in remote sensing. Zhu *et al.* [42] published a review about deep learning in remote sensing, summarizing many different neural networks.

2.2 Geo-localization

Geo-localization is the task of determining the real-world geographic location (e.g GPS coordinates) of each pixel of a query image [35].

Image-based geo-localization is the task of localizing a query ground view image on a geo-referenced satellite map [20].

2.2.1 Geo-localization for Unmanned Aerial Vehicles (UAV)

A camera is mounted on an aerial vehicle. The images are ortho-mapped as geometric correction in order to orthorectify the remote sensing data such that the scale is uniform. The localization task can then be done in various ways [36]:

- by using the inertial measurement unit (IMU) of the aerial vehicle to this way get the precise sensor position and orientation data,
- with image registration, or
- by using a number of control points (GCPs) that can then be matched with the remote sensing data.

Cross-view localization

The goal of cross-view image matching is finding its matching reference image from another view.

Cross-view localization is especially popular in autonomous driving where the camera of the vehicle is used to localize the car in the surrounding. The challenges of cross-view localization are the visual difference, the different lighting conditions and seasons and that features can be very different in cross-views [35].

2.3 Representation learning

The goal of representation or feature learning is to find a transformation that maps the data into a representation that is more suitable for machine learning tasks such as feature detection or classification tasks. Representation learning can be either supervised or unsupervised. While supervised feature learning uses labeled data, unsupervised feature learning learns the representation with unlabeled data.

2.3.1 Representation learning for Remote Sensing Data

In recent years a lot of progress in representation learning for natural and medical images was made. But according to Neumann *et al.* (2019) [27] representation learning on remote sensing data is still in an early development

stage. Reasons for this are the high diversity between remote sensing data, the need of of domain knowledge and special data processing. Also, labeling remote sensing data is expensive and therefore only recently, large-scale labeled remote sensing data is available.

The image representation for remote sensing data can be done the following ways:

- using models trained from scratch on remote sensing data,
- using models pre-trained on natural images, or
- a combination: Fine-tuning models pre-trained on natural images.

Neumann *et al.* showed that fine-tuning from ImageNet is better than training from scratch and that for small datasets fine-tuning for transfer to remote sensing data is even better [27].

2.4 Metric learning

In representation learning new spaces of representations are built. Metric learning is a specific representation learning method where a measure of distance between the data points is used as representation. In contrast to standard distance metrics like the euclidean or cosine distance, the learned distance metric is adapted to the particular data and task of interest. Popular applications of metric learning are nearest neighbors learning models for classification, clustering, information retrieval or to project the learned transformation to a new embedding space [5].

With metric learning, images can be mapped to a metric space where images of the same class are closer together than images of different classes.

2.4.1 Tile2Vec

Word2Vec is a method of processing text to represent words as vectors. Tile2Vec follows the same idea but to vectorize geospatial data. This method takes the assumption that geographic neighbours have similar semantics and accordingly images further away a different representation. It is an unsupervised representation learning method using spatial neighbourhoods with distances as form of weak supervision. The model is trained with an unsupervised triplet loss where tiles are mapped to low-dimensional embeddings [22].

2.4.2 Similarity Learning

Rather than just representing the data or mapping it to a different embedding space for clustering, similarity learning detects whether two images

represent the same object or not. For this two images are given to the model and compared. The goal is to learn a neural network that can calculate the similarity score between different images. The basic idea is that two images represent the same object if their similarity score is above a certain threshold.

In order to learn a mapping where samples of different categories can be distinguished, at least two input samples to the neural network are needed. This way the model can learn to cluster the different categories.

Siamese Neural Networks

A siamese network takes a pair of input images and processes them in parallel to transform them into an embedding space (pair of output vectors).

Here both images are processed with the same network and shared weights.

In order to learn the similarity, the distance between the embeddings is used. This distance can be represented by different loss functions where the loss is then used to further train the model. Contrastive loss and triplet loss, presented in the following section, are two commonly used loss functions.

Contrastive Loss

This loss can be used to learn to bring positive pairs together and negative pairs apart. The formula is:

$$L(x_1, x_2) = (1 - y) \cdot |f(x_1) - f(x_2)|^2 + y \cdot max(\alpha - |f(x_1) - f(x_2)|^2, 0)$$

x_1 and x_2 are the images, f the corresponding image features and y represents their ground truth relationship: if the two images are similar y is equal one, otherwise y is equal zero. The margin α is used to set the threshold on how distant dissimilar images need to be.

Triplet Neural Networks

A Triplet Neural Network takes three input samples composed by one anchor image, a positive and a negative image. The anchor and positive sample are from the same class, the negative image from a different class. The advantage of this approach is that the model can learn positive and negative distances at the same time and do a ranking as further specified in the next section.

Triplet Loss

The formula for triplet loss is:

$$L(a,p,n) = max(|f(a) - f(p)|^2 - |f(a) - f(n)|^2 + \alpha, 0)$$

a is the anchor image, p the positive image and n the negative image, f the corresponding image features. α is a margin and defines how much further apart the negative images need to be from the positive ones. The loss is calculated by the squared distance between the anchor and the positive image and squared distance between the anchor and the negative image. The idea of increasing and decreasing distances between different images is visualized in Figure 2.2.

Additionally to contrastive loss, the triplet loss can also detect how much closer a specific pair is from another pair and this way learn a ranking.

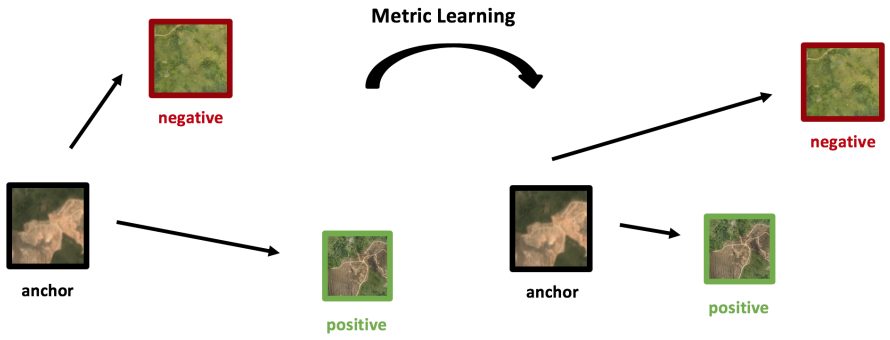


Figure 2.2: Idea of triplet loss: The distance between the anchor and positive image is decreased while the distance between the anchor and negative image is increased.

Chapter 3

Dataset

In this chapter we introduce the data, used to train, validate and test our deep learning models.

Our metric learning approach is evaluated on the following remote sensing data:

- **NAIP imagery:** The National Agriculture Imagery Program acquires aerial imagery at a one-meter ground sample distance (GSD). The data is publicly available, has four spectral bands (red, green, blue and infrared) and covers the area over the continental United States [2].
- **Sentinel-2 imagery:** Images of the Sentinel-2 satellite which is part of the Copernicus program of the European union. Sentinel-2 images are available in 13 spectral bands (including red, green and blue bands) at a 10 meter spatial resolution [3].

The NAIP and Sentinel images are collected over an area of Central Valley, California, near the city of Fesno in California, spanning latitudes [36:45; 37:05] and longitudes [-120:25;-119:65]. This is the same area as Jean *et al.* [22] use in their Tile2Vec learning approach. In total, the extracted area is about 60-by-60 km. The data is divided into 16 tiles as visualized in Figure 3.1 and further split into: 14 tiles as train set, one tile as evaluation and one tile as test set. Each tile spans an area of about 15-by-15km and is cropped to images with 200-by-200m. This results in a dataset of 2324 train images, 166 test images and 166 validation images. In this project we only work with the R,G,B channels of the images.

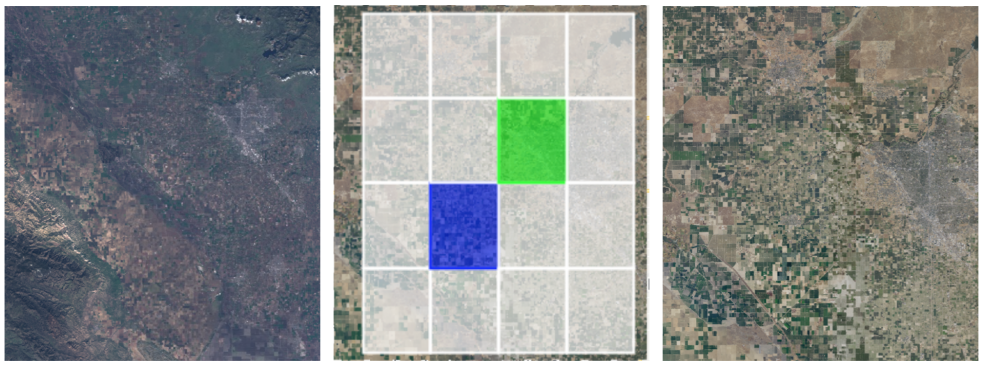


Figure 3.1: **Left:** Sentinel imagery of Central Valley, **Center:** Dataset split in train (white tiles), test (blue tile) and validation set (green tile), **Left:** NAIP imagery of Central Valley.

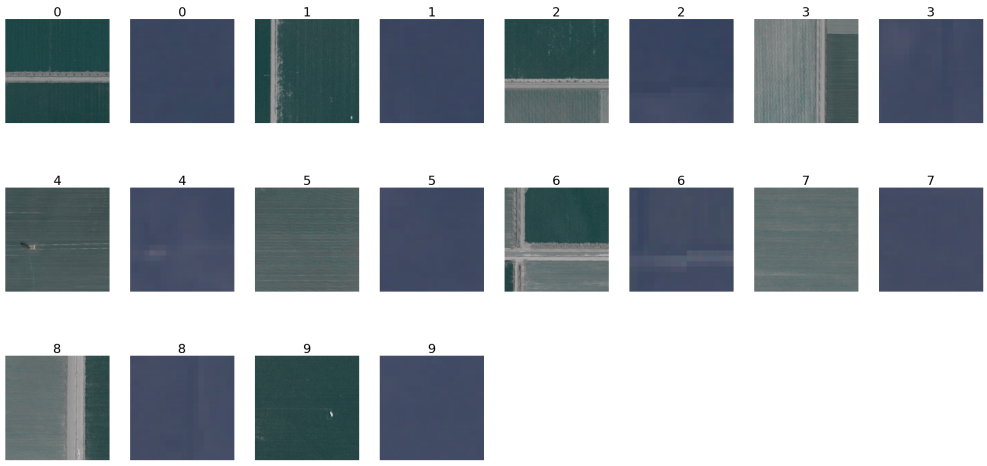


Figure 3.2: Tiles of Naip and Sentinel data for 10 different locations at 224-by-224m resolution. The images with the same label are from the same location where the first labeled image is the high-resolution aerial image and the second labeled image the corresponding low-resolution sentinel image. It can be recognized that for some locations the true image pairs seem to be more similar than for other locations.

The data was obtained from Google Earth Engine (GEE). A more detailed description is given in the Appendix A.2.

Chapter 4

Methods

In the previous chapter we introduced the data, next we present the methods used for our TrueBranch Verification System. The goal is to distinguish truthfully reported imagery from untruthfully reported imagery. The system is based on image registration where the reported drone/aerial images (at high resolution) are matched with corresponding satellite images (at low-resolution). The structure is visualized in Figure 4.1. In the introduction 1.2.2 we present possible attack vectors. In this project we focus on the distinction of locations: true pairs are images of the same location, false pairs images of different locations.

In the first section we introduce different methods for image representation model, in the second part different binary classifiers.

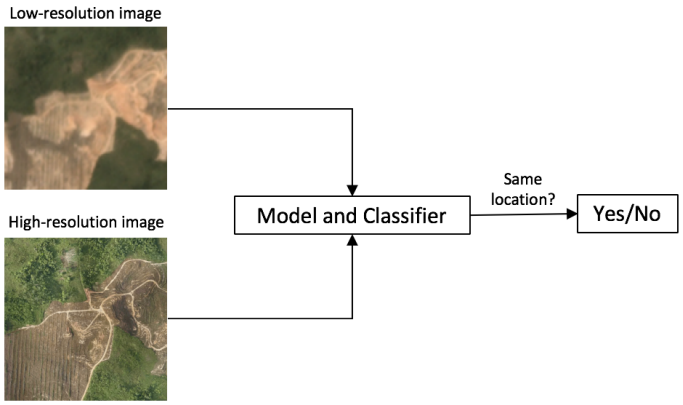


Figure 4.1: Basic flowchart of a possible forest validation algorithm. Low- and high-resolution images are given to a model and classifier which then detect whether the images are from the same location or not.

4.1 Image Representation for remote sensing data

We try three different methods to find an image representation where true and false pairs are distinguishable:

- First we compare the images using nominal metrics, comparing the images with the MSE in pixel space.
- Second we compare the images with MSE in a feature space.
- Third we use metric learning with triplet loss.

All three methods are introduced in the following section and the results presented in section 5.2.

4.1.1 Nominal distance Metrics

One method to represent images by their nominal attributes is to measure the distance between each pair of instances. This can be done in different ways.

Mean Squared Error (MSE)

A simple, commonly used method is the Mean Squared Error (MSE). The formula of the MSE between two images is:

$$MSE = \frac{1}{mn} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} [A(i,j) - B(i,j)]^2$$

n and m are the total number of pixels in x- and y-direction, nm the total number of pixels and A and B two different images. The MSE between these two images is the square of the distance between every pixel (i,j) in A and the corresponding pixel (i,j) in B, summed up for every (i,j) and divided by the total number of pixels mn.

Cosine Similarity

Another method to measure the distance between two instances is the Cosine Similarity. Here the images need to be represented as vectors. The formula is:

$$COS - Similarity = \frac{\sum_{i=0}^{n-1} A(i)B(i)}{\sqrt{\sum_{i=0}^{n-1} A(i)^2}\sqrt{\sum_{i=0}^{n-1} B(i)^2}}$$

n is the dimension of the embeddings, A and B two vectors representing two images. The Cosine similarity measures the similarity between two

images by the cosine of the angle between the two corresponding vectors. It determines how much the image representations are pointing in the same direction.

MSE in pixel space

The images are compared using the MSE in pixel space. Figure 4.2 visualises the idea of directly comparing two images with the MSE.

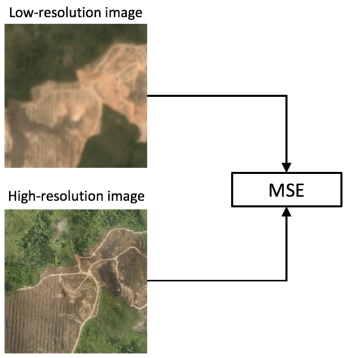


Figure 4.2: An example of directly taking the MSE in pixel space between a low- and a high-resolution image.

MSE in feature space

Instead of taking the MSE between different images in pixel space, the images are first fed into a pretrained model. The model transforms the images into a feature space, resulting in embeddings that are again be compared with the MSE. Figure 4.3 visualizes the feature space transformation and image comparison.

4.1.2 Learned Metrics

Instead of using a pretrained model to map the images into a new space and comparing them with the MSE, in learned metrics, the images are given to a model that learns a specific mapping for the given training data. In this case, low- and high-resolution remote sensing data is used to train the model. This approach is visualized in Figure 4.5 where the embeddings after the model are fit into a classifier which in this case is a MSE. The model is based on metric learning which is explained in section 2.4.

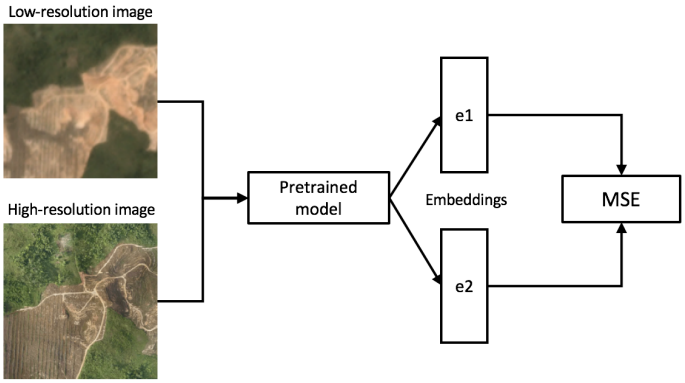


Figure 4.3: An example of taking the MSE in feature space between a low- and a high-resolution image. The images are put in the pretrained model that transforms them into a feature space, representing the images as embeddings that are then compared with the MSE.

Metric Learning with Triplet Loss

In metric learning with triplet loss, anchor, positive and negative images are selected. In our case the anchor a is the satellite image at a given location, and the positive p and negative images n and are aerial images. The triplet loss is calculated among the anchor image a, positive image p and negative image n (formula stated in section 2.4.2) and used to train the model visualized in Figure 4.4. After training, in inference, test data is fit in the model to transform the images into learned features, described by embeddings that are further used for the classification task, see Figure 4.5.

4.2 Binary Classification

The images are classified to true and false pairs by four different binary classification methods:

- First, we use a distance threshold (based on MSE or COS Similarity).
- Second, we train a Random Forest classifier on our embeddings.
- Third, we train a Logistic Regression classifier on our embeddings.
- Fourth, we train a Multilayer Perceptron classifier on our embeddings.

Section 4.2 explains the different classifiers and section 5.3 the corresponding results.

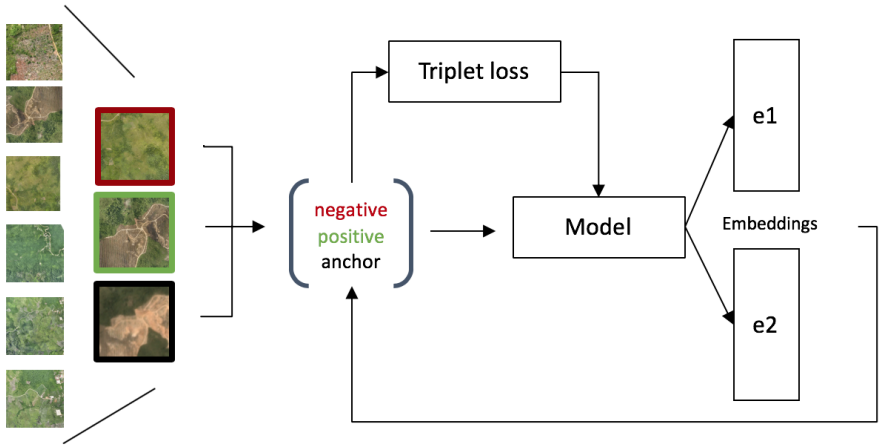


Figure 4.4: Metric learning with triplet loss - For each image an embedding is created which is then again used to calculate the triplet loss and this way optimizing the model.

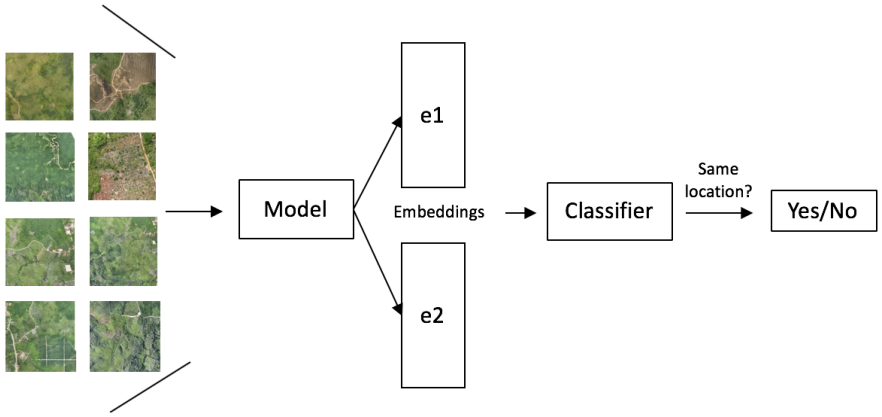


Figure 4.5: Metric learning with triplet loss - New images are fed into the trained model and this way transformed to embeddings used for classification.

The new image representation after inference is used for the classifiers to detect whether the images are from the same location or not. Furthermore, all classifiers are applied to a feature extractor baseline commonly used for comparable image classification tasks, introduced in section 5.4.

4.2.1 Threshold-based Classifier

The method is successfully able to distinguish truthful imagery from untruthful imagery if the distance between the images of different locations is significantly bigger than the distance between images of the same location.

This way a threshold can be set and the images can be classified as visualized in Figure 4.6. Figure 4.7 shows the flowchart using a trained model with a K nearest neighbors K-NN classifier.

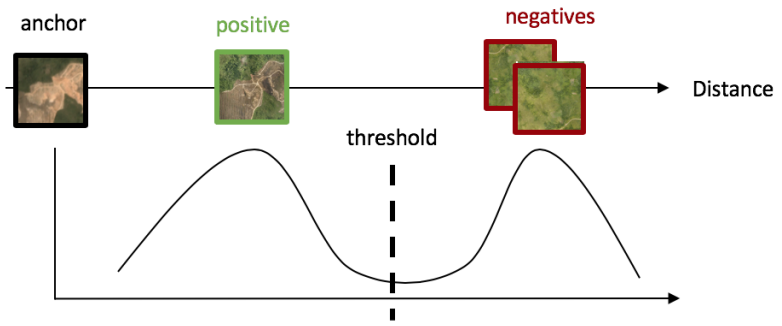


Figure 4.6: Threshold to classify images - the black framed image is the anchor at low resolution, the green framed image the positive tile (from the same location) and the red framed images the negative tile (from different locations).

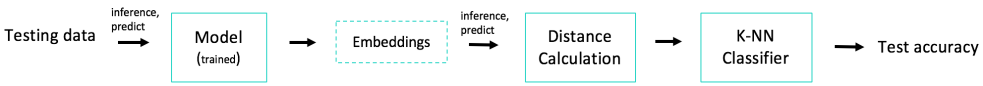


Figure 4.7: Test data is fed to the trained model to get a new representation of the images. For every anchor image, the distances to all positive and negative images is calculated. The K-NN classifier then classifies using its nearest neighbours.

Applying the same idea as in 4.6 to distinguish one location from many other locations results in a distance distribution as visualized in Figure 4.8.

4.2.2 Learned Classifiers

While the above distinction is based on the distance between the embeddings only, one can also train a classifier. Figure 4.9 shows the flowchart using a learned model and learned classifier. This workflow is chosen as it

Figure 4.8: Distances between true (marked in green) and false (marked in red) pairs. Every location has one corresponding image of the same location and many images of different locations. To distinguish the images by a threshold distance, the ture pairs need to have a smaller distance than the false pairs.

satisfies the condition that the training and test data is strictly separated and that neither the model nor the classifier is trained on test data.

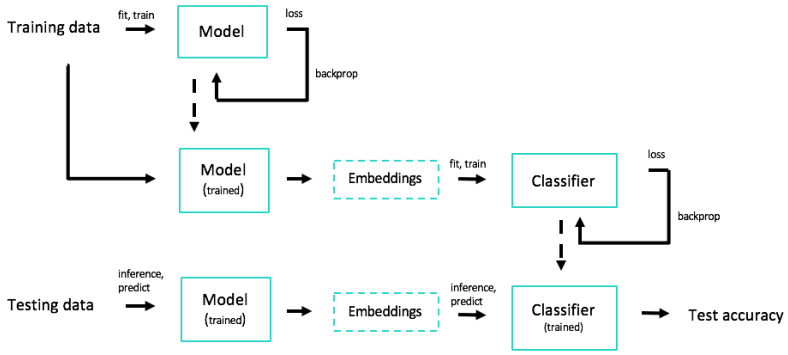


Figure 4.9: The training data is used to train the model using the loss and backpropagation. Next the same training data is given to the trained model to get its embeddings that are then further used to train the classifier. The test data is used for inference and prediction for the model and classifier, to at the end output the test accuracy.

Chapter 5

Experiments and Results

In this chapter we summarize the results obtained in our experiments. The goals of this research are to distinguish truthfully reported imagery from untruthfully reported imagery.

The chapter is structured as follows:

- In the first section we evaluate the different image representations presented in section 4.1.
- In the second section the different binary classifiers from section 4.2 are tested on our remote sensing data.
- The third section compares our results to a feature extraction baseline.

5.1 TrueBranch model architecture

Our model consists of deep residual networks ResNet-18 [18] as trunk, pretrained on ImageNet with a MLP as embedder that outputs 64-dimensional embeddings. The structure of ResNet-18 is visualized in Figure 5.1. The trunk model has an Adam Optimizer with a learning rate lr of $1e^{-5}$, the embedder model an Adam Optimizer with $lr = 1e^{-4}$. Both optimizers add a l2 penalty of $1e^{-4}$ to the cost for smaller model weights. We use the PyTorch Metric Learning implementation [26]. Musgrave *et al.* published a metric learning reality check, evaluating the performance of different metric learning algorithms and provides the corresponding libraries. The dataloader sampler is set to always include two images of the same class ($m = 2$) in each batch. This way we make sure that every batch has enough true pairs to build triplets for the loss function. We use a batch size of 64 and train over 4 epochs. The model is trained and tested as explained in section 4.1.2 on the data stated in chapter 3.

The models are implemented in Pytorch 1.4.0. The code is available on github in the climate-ai repository [1].

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Figure 5.1: Structure of a ResNet-18 convolutional neural network [18] with 18 layers, resulting in an output size of 1000.

5.2 Image Representation Evaluation

The mean squared error (MSE) is the average squared difference between two vectors. It is a normalized version of the sum of absolute difference, based on L2-norm like the euclidean distance. In the following section it is used to compare and evaluate different image representations.

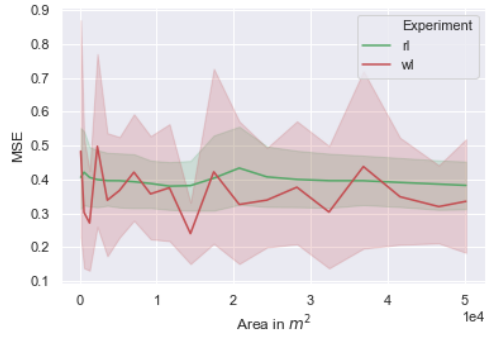
5.2.1 Analysing different area sizes

The verification performance depends on the size of the area that is compared. In this section, the MSE between images of the same location (true pairs) and images of different locations (false pairs) is calculated for varying area sizes and different image representations.

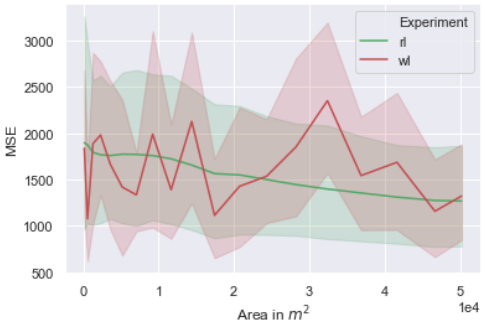
[1] https://github.com/climate-ai/truebranch

MSE in pixel space

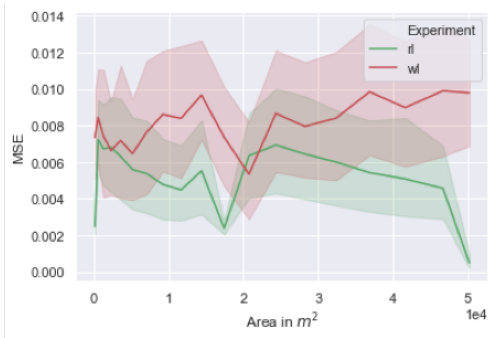
First, we try to distinguish the images with MSE in pixel space at different area sizes. The results are plotted in Figure 5.2a.



(a) MSE - Pixel space



(b) MSE - ResNet-50 feature space



(c) MSE - Metric learning

Figure 5.2: MSE for different feature spaces and different area sizes - the results show that neither the MSE metric in pixel space nor in ResNet-50 feature space are sufficient to reliably distinguish right location rl images from its attack vector wrong location wl images, whereas metric learning is able to distinguish rl-wl for an area higher than about 200-by-200m. The plots have been generated using the 20 images from Figure 3.2.

MSE in feature space

Next a ResNet-50 model, pretrained on RESISC45 remote sensing data is used [27],[8]. The obtained embeddings are again evaluated with MSE, see Figure 5.2b.

MSE without metric learning approach

Figure 5.2c shows the MSE distribution over different area sizes for our metric learning approach. Similar as for the ResNet-18 model, the images

are fit into the model and the distances between the resulted embeddings evaluated. Looking at the green and red curve, if the area of the images is enough high (at about 200mx200m), they can be distinguished.

5.2.2 Analysing model performance for 224-by-224m tiles

The above section shows that our metric learning approach should be able to distinguish different locations at an area of 224-by-224m per tile. Next, the distance from the satellite images (our anchors) to all true and false locations is calculated for all 83 location of the test data. A visualisation of the first ten locations is plotted in Figure 5.3. Applying this idea to all locations results in an accuracy of 43.4%. This means that just by looking at the MSE between true and false pairs, for 43% of all locations the true pairs have the closest locations. The Top-5 accuracy with MSE and Cosine Similarity results in 71% accuracy for both methods.

Additionally, the distance distribution of all locations is visualized in Figure 5.4.

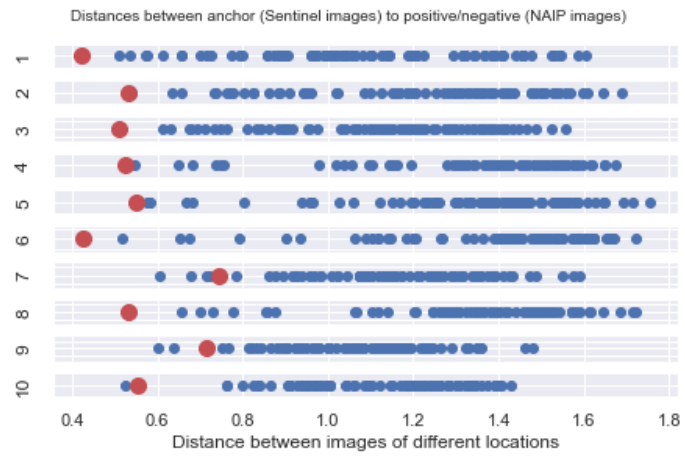


Figure 5.3: Distance between images of different locations - The y-axis shows the location, the x-axis the distance. The red point marks the same location as on the y-axis and the blue points mark different locations. Here ten locations are analyzed where our algorithm is able to distinguish the true location from the false ones for location 1, 2, 3, 4, 5, 6 and 8.

5.3 Binary Classification Evaluation

After transforming the images to a different feature space, a binary classifier is applied to classify if images are from the same location or not.

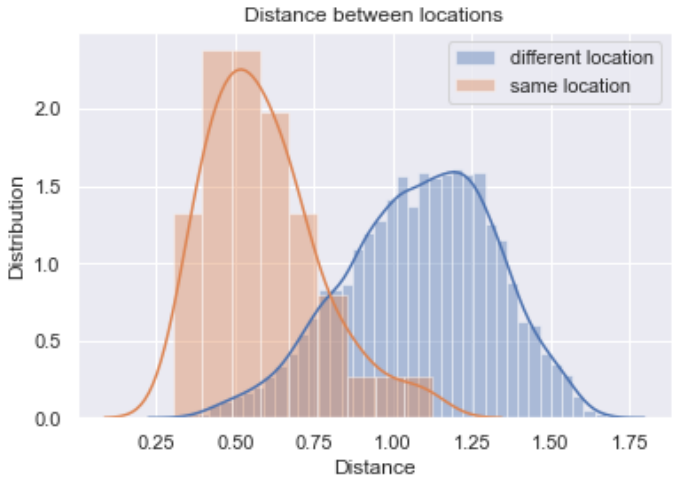


Figure 5.4: Distance distribution between all test locations - The distances between images of the same locations (true pairs) are marked in red, distances of different locations (false pairs) in blue. Comparing this graph to the threshold idea introduced in section 4.2.1, the red and blue curve show two different Gaussian distributions with an overlapping area for some locations.

5.3.1 Threshold-based Classifier

The section above shows the result of the MSE for different locations. Using this method, the MSE resulting in the highest accuracy can be set as threshold to classify the images. The same idea can be applied to Cosine similarity COS. The obtained Top-5 accuracies for our test dataset using MSE and COS as classifiers is presented in figure 5.1. If the algorithm would randomly classify the data, it would result in an accuracy of 6%, defined as the baseline.

5.3.2 Learned Classifiers

Random forest (RF), logistic regression (LR) and multilayer perceptron (MLP) classifiers are trained over 100 trials of $n = 2490$ labels (split in 2324 train and 166 test data). The train and test data are randomly paired to build two classes: one class with images from the same location and one class with images from different locations. They are sampled as two balanced classes with 1162 training and 83 test embeddings each. The mean accuracy and standard deviations are reported in Figure 5.2. The classification is a binary classification with an accuracy of 50% as baseline. The workflow of a learned model in combination with a learned classifier is visualized in Figure 4.9.

Random Forest Classifier

The random forest classifier RF is implemented with the sklearn library [29] using its default values with 100 trees in the forest.

Logistic Regression Classifier

This Logistic Regression LR classifier is implemented with the Stochastic Gradient Descent SGD approach. The method is implemented with sklearn [29], setting the loss to 'log' and the maximum number of passes over the training data to 1000.

Multi-layer Perceptron classifier

The Multi-layer Perceptron classifier MLP is implemented with sklearn [29], setting the maximum number of passes over the training data to 1000.

5.4 Feature extraction Baseline

To evaluate the performance of our model, we compare our results to a number of feature extraction methods:

- Tile2Vec: A modified ResNet-18 architecture modified for $28x28$ CIFAR-10 images, pretrained on the same area as our dataset. The extracted features have a dimension of 512.
- Pre-trained ResNet-18: A ResNet-18 model was pretrained on ImageNet and used as feature extractor. The resulted vectors have a 1000 dimensional feature space.
- Pre-trained ResNet-50: A Resnet-50 model was pretrained on Resisc-45 remote sensing data. The embeddings have a dimension of 2048.
- PCA: The dataset is reshaped from (50,50,3) images into vectors of lenght 7500. PCA is used to compute the first 10 principal components for each tile.
- K-means: Tiles are clustered in pixel space from vectors of length 7500 (3*50*50) to 1162 classes (each location is one class). The embeddings are represented as 1162-dimensional vectors of distances to each cluster centroid.
- Raw features: The tiles of shape (50,50,3) are reshaped to vectors of length 7500.

The extracted features of every method are used by Random forest (RF), logistic regression (RL) and multilayer perceptron (MLP) classifiers trained

on 2324 labels like for our metric learning method. The mean accuracy and standard deviations are reported in Figure 5.2.

Additionally the Top-5 accuracy with MSE and Cosine Similarity is calculated for all above feature extractors, see Figure 5.1.

Features	MSE based Top-5	COS based Top-5
Triplet Loss, 64-dim (ours)	67.5	67.5
Tiplet Loss, 1000-dim (ours)	**71.1**	**71.1**
Tile2Vec	8.4	8.4
Pre-trained Resnet-18 (ImageNet)	9.6	9.6
Pre-trained Resnet-50 (RESICS-45)	13.3	24.1
Raw features	25.3	45.8
PCA	42.1	34.9
K-means	3.6	4.8

Table 5.1: Comparison of our metric learning model to other feature extraction models. MSE and COS are applied on $n = 166$ labels (our test data) to calculate the Top-5 accuracy for all stated models. The feature extractors are explained in Section 5.4.

Features	RF	LR	MLP
Triplet Loss, 64-dim (ours)	83.9±2.0	49.7±2.0	**81.9±2.1**
Tiplet Loss, 1000-dim (ours)	**85.1±2.3**	50.2±1.9	80.0±7.0
Tile2Vec	60.0±2.8	50.1±1.3	56.12±2.8
Pre-trained Resnet-18 (ImageNet)	66.6±3.2	50.0±0.4	50±1.4
Pre-trained Resnet-50 (RESICS-45)	77.6±2.9	50.4±3.0	80.3±2.5
Raw features	62.1±3.0	50.0±0.4	50.5±1.7
PCA	50.0±3.7	49.5±1.8	50.0±1.3
K-means	62.0±3.0	49.9±0.6	50.4±1.5

Table 5.2: Comparison of our metric learning model to other feature extraction models. Random forest (RF), logistic regression (LR), and multilayer perceptron (MLP) classifiers are trained over 100 trials of $n = 2490$ labels, with mean accuracies and standard deviations reported. The feature extractors are explained in Section 5.4.

The Nearest Neighbor algorithm performs best with 71% Top-5 accuracy, compared to a baseline of 6%. The Learning based Algorithm performs best with RF, achieving an accuracy of 85%, compared to a baseline of 50%. The Nearest Neighbor algorithm does not need training data, is fast in runtime and performs well compared to its baseline. In practice, we therefore

suggest to use the Nearest Neighbor algorithm as classifier.

Chapter 6

Conclusion and Future Work

Landowners can get financial support for their trees but for this a transparent MRV is necessary. To verify that the reported high-resolution drone images are from the correct location, we use low-level satellite images. We implemented a model and classifier that tells us whether the drone and satellite images are from the same location or not.

We trained several models (see section 5.4 on our dataset, evaluating their performance of detecting whether images are from the same location or not. The results 5.1 show that our embeddings trained with metric learning perform best for our task.

The Top-5 accuracy is 71%. This accuracy decreases the incentive of submitting wrong drone images. If a landowner reports wrong images, the classifier detects every wrong image with a chance of 71%. Thus there is a high chance of getting caught. Furthermore, the algorithm outputs the corresponding satellite images. This way, a human can additionally verify if by-eye the drone image corresponds to the satellite image. This will additionally increase the performance of TrueBranch verification, reliably detecting images from the same and different locations.

We submitted a proposal paper [31] to the ICLR 2020 Workshop on Tackling Climate Change and won the best proposal paper award. There we suggested metric learning with triplet loss. These results are promising and a motivation to publish a paper.

To further increase the accuracy we suggest the following tasks:

- Remote sensing data includes an infrared channel. We suggest to use this additional information in future work.
- The model should be further tuned with hyperparameter optimization.

- We used data from California, future work should include additional datasets from different regions. This will increase the accuracy and increase fairness as the algorithm is trained on different areas.

To further increase the robustness we suggest the following tasks:

- In this project we focused on detecting if the images are from the same location or not. In section 1.2.2 we present different attack vectors. TrueBranch should be protected against all possible attack vectors.
- The landowner reports private information such as his GPS data and images of his land. A privacy preserving analysis should assure that his data is hold private.

Appendix A

Data Collection

Satellite images (here from Sentinel) can be collected the following way from gee:

Listing A.1: GEE Satellite Data Collection

```
/**
 * Function to mask clouds using the Sentinel-2 QA band
 * @param {ee.Image} image Sentinel-2 image
 * @return {ee.Image} cloud masked Sentinel-2 image
 */

function maskS2clouds(image) {
  var qa = image.select('QA60');

  // Bits 10 and 11 are clouds and cirrus, respectively
  var cloudBitMask = 1 << 10;
  var cirrusBitMask = 1 << 11;

  // Both flags zero, indicating clear conditions
  var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
      .and(qa.bitwiseAnd(cirrusBitMask).eq(0));

  return image.updateMask(mask).divide(10000);
}

var area= ee.Geometry.Rectangle(-120.25, 36.45, -119.65,37.05)

// Load Sentinel-2 TOA reflectance data.
var dataset = ee.ImageCollection('COPERNICUS/S2')
                  .filterDate('2016-01-01', '2016-12-31')
```

```
                    .filterBounds(area)
                    // Pre-filter to get less cloudy granules.
                    .filter(ee.Filter.lt(
                    'CLOUDY_PIXEL_PERCENTAGE',20))
                    .map(maskS2clouds);

var mosaic = dataset.select(['B4', 'B3', 'B2']).mosaic()

var rgbVis = {
  min: 0.0,
  max: 0.3,
  bands: ['B4', 'B3', 'B2'],
};

Map.setCenter(-119.71, 36.45, 10);
Map.addLayer(mosaic, rgbVis, 'RGB');

var Sentinel_RGB = mosaic.visualize({bands:
['B4', 'B3', 'B2'], min: 0.0, max: 0.3});

Export.image.toDrive({
  image: Sentinel_RGB,
  description: 'Sentinel_image',
  scale: 10,
  region: area,
  maxPixels: 2000000000
});
```

Aerial images from NAIP can be collected the following way from gee:

Listing A.2: GEE NAIP Data Collection

```
var area = ee.Geometry.Rectangle(-120.25, 36.45, -119.65,37.05)

// Load four 2012 NAIP quarter quads, different locations.
var naip2016 = ee.ImageCollection('USDA/NAIP/DOQQ')
  .filterBounds(area)
  .filterDate('2016-01-01', '2016-12-31');

// Spatially mosaic the images in the collection and display.
var naip_mosaic = naip2016.mosaic();

Map.setCenter(-119.71,36.45, 10);
Map.addLayer(naip_mosaic, {}, 'spatial␣mosaic');
```

```
Map.addLayer(area, {color: 'FF0000'}, 'geodesic polygon');

var Naip_RGB = naip_mosaic.visualize({bands:
['B4', 'B3', 'B2']});

// Export a cloud-optimized GeoTIFF.
Export.image.toDrive({
  image: Naip_RGB,
  description: 'NAIP_image',
  scale: 1,
  region: area,
  fileFormat: 'GeoTIFF',
  formatOptions: {
    cloudOptimized: true
  }
});
```

Listing A.3: Extract a specific tile from a big region

```
def extract_tile(mosaicdb, lon, lat, tile_size, crs):
    '''
    Extract tile_size x tile_size tile
    from a tif starting from lon, lat
    mosaicdb: Rasterio Data Reader
    '''
    assert crs in ['ESPG:3857', 'ESPG:4326']
    if crs == 'ESPG:4326':
        xgeo, ygeo = geodesic2spherical(lon, lat)
    else:
        xgeo, ygeo = lon, lat
    idx, idy = mosaicdb.index(xgeo, ygeo)
    return mosaicdb.read(window=Window(idy, idx, 256, 256))
```

Listing A.4: Generate Hansen Tile

```
def gen_tile(img_db, lon, lat, tile_type, year, out_name):
    """
    Generates a 256x256 forest cover or forest loss tile
    starting from (lon, lat).
    """
    assert tile_type in ['ld', 'fc', 'fl']
    FC_IDX = 0 # forest cover index
    GAIN_IDX = 1 # forest gain index
    LOSS_IDX = 2 # forest loss index
    img_arr = extract_tile(img_db,
```

```
            lon, lat, 256,
            crs='ESPG:4326')

        if tile_type == 'fc':
            loss_arr = np.copy(img_arr[LOSS_IDX])
            loss2000_2012 = get_aggregated_loss(loss_arr,
                beg=1, end=12)
            gain2000_2012 = np.copy(img_arr[GAIN_IDX])
            loss2013_year = get_aggregated_loss(loss_arr,
                beg=13, end=year)
            fc2000 = np.copy(img_arr[FC_IDX])
            fc2000 = preprocess_fc(fc2000)
            img_arr = create_forest_cover(fc2000,
    gain2000_2012,
                loss2000_2012,
                loss2013_year)

        elif tile_type == 'fl':
            lon, lat, 256, crs='ESPG:4326')
            img_arr = np.copy(img_arr[LOSS_IDX])
            loss_mask = np.where(img_arr == year)
            no_loss_mask = np.where(img_arr != year)
            img_arr[loss_mask] = 1
            img_arr[no_loss_mask] = 0

        np.save(out_name, img_arr)
```

List of Tables

List of Figures

Bibliography

[1] Interviews with ministry of agriculture peru.

[2] Naip. https://www.fsa.usda.gov/programs-and-services/aerial-photography/imagery-programs/naip-imagery/. Accessed: 2020-11-06.

[3] Sentinel. https://sentinel.esa.int/web/sentinel/missions/sentinel-2. Accessed: 2020-11-06.

[4] Truebranch, github repository. https://github.com/climate-ai/truebranch. Accessed: 2020-11-06.

[5] What is metric learning. http://contrib.scikit-learn.org/. Accessed: 2020-11-06.

[6] Ali Ismail Awad and Mahmoud Hassaballah. Image feature detectors and descriptors. *Studies in Computational Intelligence. Springer International Publishing, Cham*, 2016.

[7] Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert DiBiano, Manohar Karki, and Ramakrishna Nemani. Deepsat: a learning framework for satellite imagery. In *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, pages 1–10, 2015.

[8] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[9] Mohamed Dahmane, Samuel Foucher, Mario Beaulieu, Yacine Bouroubi, and Mathieu Benoit. The potential of deep features for small

object class identification in very high resolution remote sensing imagery. In *International Conference Image Analysis and Recognition*, pages 569–577. Springer, 2017.

[10] Mohamed Dahmane, Samuel Foucher, Mario Beaulieu, F Riendeau, Yacine Bouroubi, and Mathieu Benoit. Object detection in pleiades images using deep features. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1552–1555. IEEE, 2016.

[11] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016.

[12] Stephen Donofrio, Patrick Maguire, William Merry, and Steve Zwick. Financing emissions reductions for the future. In *State of the Voluntary Carbon Markets 2019*. Ecosystem Marketplace; A Forest Trends Initiative, 2019.

[13] Orhan Firat, Gulcan Can, and Fatos T Yarman Vural. Representation learning for contextual object and region detection in remote sensing. In *2014 22nd International Conference on Pattern Recognition*, pages 3708–3713. IEEE, 2014.

[14] Holly K Gibbs, Sandra Brown, John O Niles, and Jonathan A Foley. Monitoring and estimating tropical forest carbon stocks: making REDD a reality. *Environmental Research Letters*, 2(4):045023, 2007.

[15] Gold Standard. Afforestation/reforestation (a/r) ghg emissions reduction & sequestration methodology, July 2017.

[16] A Ardeshir Goshtasby. *Image registration: Principles, tools and methods*. Springer Science & Business Media, 2012.

[17] M. C. Hansen, P. V. Potapov, R. Moore, M. Hancher, S. A. Turubanova, A. Tyukavina, D. Thau, S. V. Stehman, S. J. Goetz, T. R. Loveland, A. Kommareddy, A. Egorov, L. Chini, C. O. Justice, and J. R. G. Townshend. High-resolution global maps of 21st-century forest cover change. *Science*, 342(6160):850–853, 2013.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Noriko Hosonuma, Martin Herold, Veronique De Sy, Ruth S De Fries, Maria Brockhaus, Louis Verchot, Arild Angelsen, and Erika Romijn. An assessment of deforestation and forest degradation drivers in developing countries. *Environmental Research Letters*, 7(4):044009, 2012.

[20] Sixing Hu and Gim Hee Lee. Image-based geo-localization using satellite imagery. *International Journal of Computer Vision*, pages 1–15, 2019.

[21] IPCC. 2019: Summary for policymakers. In P.R. Shukla, J. Skea, E. Calvo Buendia, V. Masson-Delmotte, H.-O. Pörtner, D. C. Roberts, P. Zhai, R. Slade, S. Connors, R. van Diemen, M. Ferrat, E. Haughey, S. Luz, S. Neogi, M. Pathak, J. Petzold, J. Portugal Pereira, P. Vyas, E. Huntley, K. Kissick, M. Belkacemi, and J. Malley, editors, *Climate Change and Land: an IPCC special report on climate change, desertification, land degradation, sustainable land management, food security, and greenhouse gas fluxes in terrestrial ecosystems*, pages 7–11. 2019.

[22] Neal Jean, Sherrie Wang, Anshul Samar, George Azzari, David Lobell, and Stefano Ermon. Tile2vec: Unsupervised representation learning for spatially distributed data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3967–3974, 2019.

[23] Julian Krebs, Tommaso Mansi, Hervé Delingette, Li Zhang, Florin C Ghesu, Shun Miao, Andreas K Maier, Nicholas Ayache, Rui Liao, and Ali Kamen. Robust non-rigid registration through agent-based action learning. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 344–352. Springer, 2017.

[24] Quoc V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013.

[25] Rui Liao, Shun Miao, Pierre de Tournemire, Sasa Grbic, Ali Kamen, Tommaso Mansi, and Dorin Comaniciu. An artificial agent for robust image registration. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[26] Kevin Musgrave, Ser-Nam Lim, and Serge Belongie. Pytorch metric learning. https://github.com/KevinMusgrave/pytorch-metric-learning, 2019.

[27] Maxim Neumann, Andre Susano Pinto, Xiaohua Zhai, and Neil Houlsby. In-domain representation learning for remote sensing. *arXiv preprint arXiv:1911.06721*, 2019.

[28] Ty Nguyen, Steven W Chen, Shreyas S Shivakumar, Camillo Jose Taylor, and Vijay Kumar. Unsupervised deep homography: A fast and robust homography estimation model. *IEEE Robotics and Automation Letters*, 3(3):2346–2353, 2018.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] Otávio AB Penatti, Keiller Nogueira, and Jefersson A Dos Santos. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 44–51, 2015.

[31] Simona Santamaria, David Dao, Björn Lütjens, and Ce Zhang. Truebranch: Metric learning-based verification of forest conservation projects, 2020.

[32] Moslem Ouled Sghaier, Imen Hammami, Samuel Foucher, and Richard Lepage. Stroke width transform for linear structure detection: Application to river and road extraction from high-resolution satellite images. In *International Conference Image Analysis and Recognition*, pages 605–613. Springer, 2017.

[33] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[34] Mohamed Tahoun, Abdelrahman Shabayek, Marcello Giovenco, Eid Emary, Ralf Reulke, Hamed Nassar, and Aboul Ella Hassanien. *Satellite Image Matching and Registration: A Comparative Study Using Invariant Local Features*, volume 630, page 37. 02 2016.

[35] Yicong Tian, Chen Chen, and Mubarak Shah. Cross-view image matching for geo-localization in urban environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3616, 2017.

[36] Chung-Hsien Tsai and Yu-Ching Lin. An accelerated image matching technique for uav orthoimage registration. *ISPRS Journal of Photogrammetry and Remote Sensing*, 128:130–145, 2017.

[37] Fabien Hubert Wagner, Matheus Pinheiro Ferreira, Alber Sanchez, Mayumi CM Hirye, Maciel Zortea, Emanuel Gloor, Oliver L Phillips, Carlos Roberto de Souza Filho, Yosio Edemir Shimabukuro, and Luiz EOC Aragão. Individual tree crown delineation in a highly diverse tropical forest using very high resolution satellite images. *ISPRS journal of photogrammetry and remote sensing*, 145:362–377, 2018.

[38] Aled Williams. Using corruption risk assessments for redd+: An introduction for practitioners. *U4 Issue*, 2014.

[39] Sven Wunder. The efficiency of payments for environmental services in tropical conservation. *Conservation Biology*, 21(1):48–58, 2007.

[40] Yi Yang and Shawn Newsam. Geographic image retrieval using local invariant features. *IEEE Transactions on Geoscience and Remote Sensing*, 51(2):818–832, 2012.

[41] X. X. Zhu, D. Tuia, L. Mou, G. 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*, 5(4):8–36, 2017.

[42] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine*, 5(4):8–36, 2017.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
	S. Santaruccia

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.