University of Münster

Department of Information Systems

Advancements in Remote Sensing: Developing an Updated Forest Mask through Deep Learning

Bachelor Thesis

submitted by

Julian Sibbing

Chair of Data Science:
Machine Learning and Data Engineering

Principal Supervisor	Prof. Dr. Fabian Gieseke
Supervisor	Jan Pauls, M.Sc.
	Chair for Data Science:
	Machine Learning and
	Data Engineering
Student Candidate	Julian Sibbing
Matriculation Number	524435
Field of Study	Information Systems
Contact Details	jsibbing@uni-muenster.de
Submission Date	04.08.2024

Contents

1 Introduction

Forests contain more than 80% of living biomass [SBP05, p. 587], allowing them to store large amounts of emitted carbon [Pan+11]. Storage of CO_2 is one element of the roadmap towards minimizing CO_2 emissions [YD20, p. 6f.]. Minimizing CO_2 emissions is crucial, as a rising concentration of greenhouse gas (GHG) emissions in the atmosphere leads to global warming and climate change [YD20, p. 4]. As a result of climate change, global agricultural production is declining due to reduced rainfall, seasonal variability, and rising temperatures, making many parts of the world unsuitable for commercial agriculture due to increased drought [SY16; YD20].

Consequently, resources to monitor and manage forests as a key ecosystem to mitigate the effects of climate change are urgently needed.

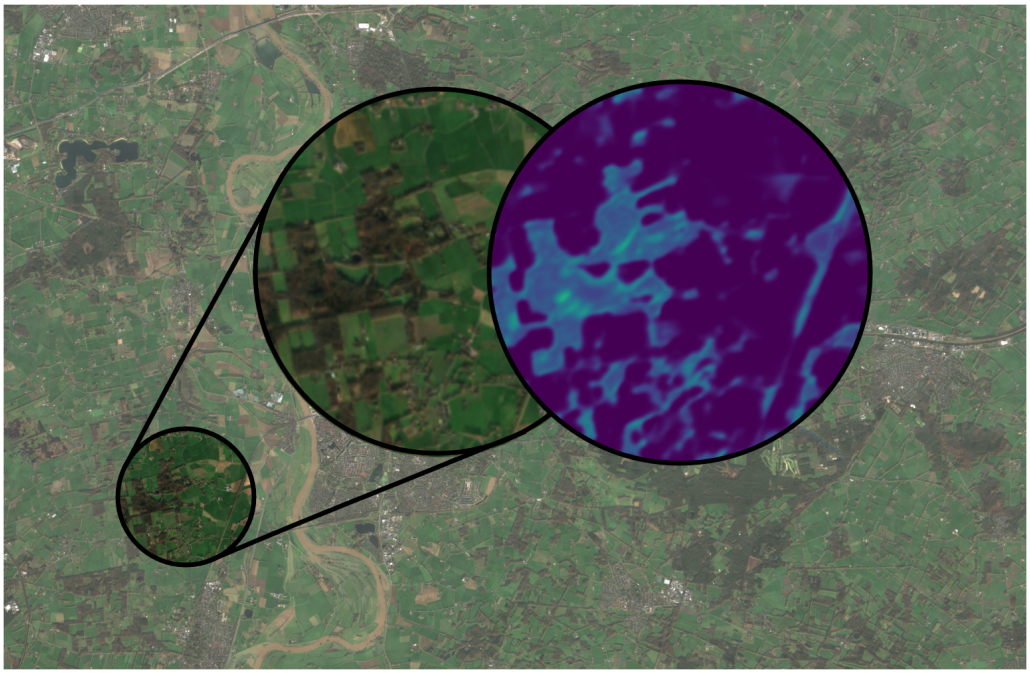


Figure 1 Forest mask on 10m resolution Sentinel-2 imagery

In the past, historical maps were the main resource for spatial information about land cover and land use [Mun+14; BSG15]. Comparisons between these maps help to study changes in land cover. This process, however, is very time-consuming and not applicable at scale [SIN89; Yan+14]. With the recent availability of frequent remote sensing (RS) data, analyzing the earth's surface has become easier. Instead of comparisons, time-series analysis led to scalable results of higher quality [Ban+14].

One important application of RS in forest management is the creation of *forest masks*. Forest masks are forest cover segmentation maps [Van+21]. These maps are useful

to monitor the ecosystem and manage it accordingly. Moreover, forest masks might be used to filter existing maps, like canopy height maps [Pau+24], to estimate the forest biomass [Cha+14].

One global forest mask is that of Hansen et al. [Han+13]. It shows the tree cover, depicted as the canopy closure of all vegetation taller than 5 meters, in the year 2000 with a resolution of 30 meters. However, the improvable resolution, which results in lower-quality results, in addition to the outdated data, creates the need for newer, improved forest masks.

Hansen et al.'s forest mask was obtained using a time series analysis. More specifically, it has been created through the aggregation of data from different timestamps (mulitdate composition) and the implementation of a decision tree classifier [Ban+14]. This approach is unprecedented, because machine learning (ML) techniques, which involve training algorithms to recognize patterns and make predictions based on data, were commonly used during the time of creation [Ma+19]. Since then, deep learning (DL), a subset of ML that utilizes neural networks with many layers to model complex patterns in data, has been widely used within RS [Zhu+17; Li+18; Ma+19]. The efficacy of DL has been demonstrated in the domain of computer vision and, subsequently, in RS as well [LBH15].

Other than advancements in techniques applied in RS, the data available has also seen improvement. Hansen et al. used data provided by Landsat-1, "the first Earth-observing satellite to be launched with the express intent to study and monitor our planet's landmasses"[1]. Today, many more space programs are dedicated to collecting data about Earth. Namely, the Copernicus[2] program of the European Union (EU). Copernicus' Sentinel-2 mission[3] uses two orbiting satellites to produce data about the earth's surface with more frequency and higher resolution than Landsat-1.

Using the recent advancements within the field of RS, the research goal of this thesis is the *development of an updated forest mask for Germany* of 10-meter resolution using Sentinel-2 data with labels from Hansen et al. [Han+13] (see Figure 1).

The remaining thesis is outlined as follows: Section 2 revisits the literature for the fields of RS (Section 2.1) and DL (Section 2.2), which are necessary as a background for the remaining thesis. Section 3 will give insight into the data used to fulfill the research goal. Section 4 explains the methods used to approach the goal. The presentation of results ranges from Sections 5-8. Section 5 will provide all the necessary means to develop a base model capable of deriving the first forest mask. Section 6

1 `https://landsat.gsfc.nasa.gov/satellites/landsat-1/`
2 `https://www.copernicus.eu/en/about-copernicus`
3 `https://copernicus.eu/copernicus/sentinel-2`

will introduce the Many-To-One U-Net, a novel modification to the U-Net, capable of processing more spatial information. Section 7 will clarify challenges with the data and introduce a regularization strategy applied to address these challenges. In Section 8, a superordinate final comparison of the results from the previous experiments is carried out. Section 9 will put the results into the research context and explain possible drawbacks. Lastly, Section 10 will give a comprehensive overview of the thesis.

Table 1 Notation

X, $X^{(tr)}$, $X^{(val)}$, $X^{(te)}$	Dataset, training set, validation set, test set
x	Sample features
y	label / ground truth
$\hat{y}$	label prediction (one-hot vector)
$\hat{p}$	probability prediction (softmax output)
$\mathcal{L}$	Loss function
α	Activation function
λ	Hyperparameter
θ	Model weights
$\mathcal{M}_\theta$	Model

2 Research Background

This section provides a comprehensive overview of the key concepts and advancements in the field of remote sensing and deep learning, outlining the resources utilized for the research goal.

2.1 Remote Sensing

RS has been defined many times. Campbell and Wynne define RS broadly as the "gathering of information at a distance" [CW11, p. 4]. They go on to define it more precisely as "the practice of deriving information of Earth's land and water surfaces using images acquired from an overhead perspective, using electromagnetic radiation in one or more regions of the electromagnetic spectrum, reflected or emitted from the Earth's surface" [CW11, p. 6].

In RS, the detection and discrimination of objects or surface features involve the process of capturing and measuring the radiant energy that is reflected or emitted by these objects or materials. This radiant energy is part of the electromagnetic spectrum, which includes a range of wavelengths from visible light to infrared wavelengths that are invisible to the human eye [AD05].

The necessary data is typically obtained using satellites, which have been used to repeatedly examine Earth's whole surface since 1972 with the launch of *Landsat 1* [CW11, p. 15]. Another prominent source of data are unmanned aerial vehicles (UAV), commonly referred to as *drones*. Their lower cost and more flexible deployment are an advantage over satellites [Osc+21].

RS provides essential means for a wide variety of applications. Examples include (i) sustainable agriculture and (ii) disaster monitoring.

(i) The Crop Acreage and Production Estimate (CAPE) project estimates the pre-harvest crop area on a district level and further forecasts the harvest using remote-sensing data [Dad+02].

(ii) The Disaster Management Support Program (DMSP) combines different remote data sources covering flood monitoring and land use to identify endangered settlements and support measures or assess potential damage [NVR07].

Advances in technology allow satellites to obtain large volumes of data, so much so that a single satellite data center gathers several terabytes of data every day [Ma+15].

The introduction of big data in the field of RS enables the use of DL to gain valuable insights from the data [Zho+17].

Prior to applying DL in the field, ML techniques like support vector machines (SVM) [CV95] and random forests (RF) [Ho95] received much attention for various tasks, including image classification [Ma+19].

Starting in 2014, successes in recognition, object detection and semantic segmentation on large-scale images have led the use of DL to take off within the RS community [Zhu+17; Li+18; Ma+19].

Zhang, Zhang, and Du concluded that "DL is actually everywhere in RS data analysis: from the traditional topics of image preprocessing, pixel-based classification and target recognition, to the recent challenging tasks of high-level semantic feature extraction, and RS scene understanding" [ZZD16].

2.2 Deep Learning

Goodfellow, Bengio, and Courville defined DL as systems "that allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to simpler concepts" [GBC16, p. 1]. DL is part of ML, the category for techniques that improve from experience and data. The unique aspect of DL is its assumption of a hierarchical representation of the environment. Artificial intelligence (AI) is even broader and also includes rule-based systems, like simple chatbots (take ELIZA [Wei66] as an example) that do not improve with experience [Ong17]. Figure 2 provides an overview of the relationships.

DL uses a multilayered network to process an input given through an *input layer* in order to process a given task's result within the *output layer*. The number of processing layers between the input and output layers, which are known as *hidden layers*, can be quite extensive. Hence, the name, *deep learning*. Each layer in the network is composed of nodes (*neurons*) connected to the neurons of the prior and following layers by connections, to which a weight is applied.

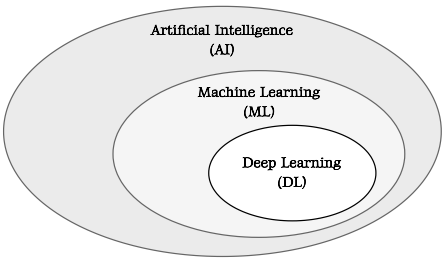


Figure 2 Definitions within the AI landscape showing DL as a subdiscipline of ML [GBC16, p. 9]

A popular example to illustrate the hierarchical abstractions deep neural networks (DNN) learn is handwriting recognition with the MNIST dataset [LeC+98; Den12]. A dataset containing 60 000 of handwritten images of the digits 0 – 9. Deriving from Goodfellow et al.'s definition, the DNN learns the digit representation through simpler concepts. Meaning that a first layer might be interested in the presence or absence of edges or curves. A second layer examines composites like crosses or circular shapes [LBH15]. The network will adjust to the representations that help it classify the handwritten images the best by minimizing the loss function.

Tasks performed by DNN, other than classification, include *semantic segmentation*. Semantic segmentation is described as a next step from coarse to fine inference [Gar+17]. According to Garcia-Garcia et al., its goal is to "make dense predictions inferring labels for every pixel; this way, each pixel is labeled with the class of its enclosing object or region" [Gar+17].

2.2.1 Supervised Learning

LeCun, Bengio, and Hinton state that "the most common form of ML, deep or not, is supervised learning (SL)" [LBH15]. In a SL scenario, each item in the dataset is paired with the ground truth, also referred to as the target [GBC16, p. 103]. The training process is designed as follows:

Feed-forward

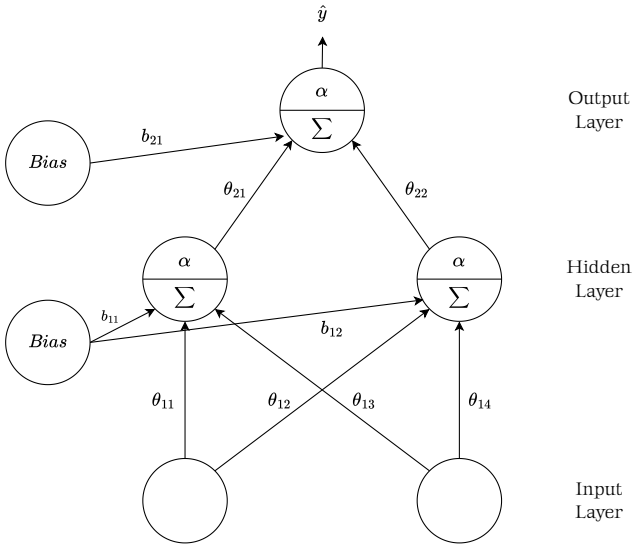


Figure 3 Architecture of a feedforward Neural Network

In the training process, the input data is fed forward through the network. In the feed-forward **1.** each neuron receives the weighted signal $\theta_{jk}y_j$ from the prior neurons. j denotes the previous layer's neurons and k the current layer's neurons. **2.** computes their sum $z_k = \sum_{j \in H} \theta_{jk}y_j + b_k$ with H being the prior hidden layer, b_k a bias, and **3.** determines the neuron's activation using an activation function α by computing $\alpha(z_k)$. By forwarding the input through the whole DNN this way, the model's predictions resemble those in the output layer. Figure 3 shows an exemplary architecture of a feedforward neural network.

Neural networks (NN) do not necessarily need to be feedforward networks; the category of NN that implements a feedback loop within the network constitutes the field of *recurrent neural networks* (RNN) [GBC16, p. 164].

Error determination

The output of the model (e.g., in a classification task, a vector with probabilities for each class) is then compared to the ground truth using a loss function. In classification, the cross entropy (CE) is typically used [HB21]. It is shown in Equation 2.1 [Tia+22], with y being the ground truth and $\hat{p}$ the network output. As the ground truth is a one-hot vector, meaning one dimension (the true class) is 1 with the rest being 0, only the calculation in the positive dimension affects the result. Because in that dimension $y_j = 1$, the loss can be put as $-log(\hat{p_j})$ with regard to j as the dimension of the target class. The logarithm exponentially increases the penalty for predictions of the true class that are further from 1 (100%).

$$\mathcal{L}_{\mathrm{CE}}(y, \hat{p}) = -\sum_j y_j \cdot \log \hat{p_j} \tag{2.1}$$

In tasks identifying a single class (e.g., forest or no forest), the Binary Cross Entropy (BCE) (Equation 2.2) [HW20] is applied. BCE additionally penalizes high confidence in the wrong class [HW20].

$$\mathcal{L}_{\mathrm{BCE}}(y, \hat{p}) = -\sum_j y_j \cdot \log \hat{p_j} + (1 - y_j) \cdot \log(1 - \hat{p_j}) \tag{2.2}$$

Backward Pass

The backward pass is typically done using backpropagation. Backpropagation is the most popular learning rule for supervised learning [Hec89; Wer90; LH91; Rum+95]. With backpropagation, the effect of each weight on the loss function is recursively *propagated back* through the network, starting from the output layer. Formally speak-

ing, backpropagation computes the partial derivative $\frac{\delta \mathcal{L}}{\delta \theta_{ij}}$ for each weight [Wer90]. The chain rule simplifies the differential (see Equation 2.3) [GBC16, p. 203ff.] with z being the weighted sum of the neuron's input as described in the forward pass.

$$\frac{\delta \mathcal{L}}{\delta \theta_{ij}} = \frac{\delta \mathcal{L}}{\delta a_j} \frac{\delta a_j}{\delta z_j} \frac{\delta z_j}{\delta \theta_{ij}} \tag{2.3}$$

Gradient Descent

Using the gradient derived from backpropagation, the gradient step performs the update on the actual weight [Rud17]. The gradient's property to show the steepest slope is used to minimize the error (go in the opposite direction). The basic formula is given in Equation 2.4 [Rud17] with η being a factor by which the old weights are updated.

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla \mathcal{L}(\theta_t) \tag{2.4}$$

Three different types of gradient descent are differentiated depending on the data on which the gradient is computed. Stochastic gradient descent (SGD) works by computing the gradient step for each training sample, as shown in Equation 2.5 [Rud17]. In contrast, regular gradient descent computes the step for the whole batch [Rud17]. While SGD is quicker than regular gradient descent, it lacks in performance. *Mini-batch* gradient descent strikes a balance between performance and speed by using a subset of the entire training set for each iteration [Rud17].

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla \mathcal{L}(\theta_t; x^{(i)}; y^{(i)}) \tag{2.5}$$

The basic algorithm is no longer used in practice [Qia99]. More evolved optimizers are introduced in the following.

RMSProp. By adapting the learning rate for each weight dimension, RMSProp (Equation 2.6 [HT12]) can find a more efficient trajectory to local and global minima. A moving average of the squared gradient for each weight is used for the adapted learning rate [HT12]. The idea is that dimensions with large steps in each iteration are adjusted proportionally to the other dimensions. s keeps the moving squared gradient averages parameterized by β. Hinton and Tieleman recommend using $\beta = 0.9$ [HT12]. ϵ assures that the value beneath the root is non-zero and is therefore kept at a small value, typically 10^{-8} [Cho+20].

$$
\begin{aligned}
s_t &= \beta s_{t-1} + (1-\beta) \cdot \nabla \mathcal{L}(\theta_t) \odot \nabla \mathcal{L}_\theta(\theta_t) \\
\theta_{t+1} &= \theta_t - \eta \cdot \nabla \mathcal{L}(\theta_t) \sqrt{s_t + \epsilon}
\end{aligned}
\tag{2.6}
$$

Adam. Adam (Equation 2.7 [KB17]) has become the standard optimizer for DL applications [Wil+18]. Similar to RMSProp, Adam uses adaptive learning rates for the different weights. Other than keeping the moving average of squared gradients s, it also uses the moving average of gradients m. The addition of m introduces a component of similar to momentum [Qia99]. Due to their initialization as vectors of zeros, the moving averages are biased toward zero at the beginning of training, so bias-corrected estimates $\hat{m}$ and $\hat{s}$ are calculated [KB17].

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1) \cdot \nabla \mathcal{L}(\theta_t) \\
s_t &= \beta_2 m_{t-1} + (1-\beta_2) \cdot \nabla \mathcal{L}(\theta_t) \odot \nabla \mathcal{L}(\theta_t) \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{s}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_{t+1} &= \theta_t - \eta \cdot \hat{m}_t \oslash \sqrt{\hat{s}_t + \epsilon}
\end{aligned}
\tag{2.7}
$$

Gradient descent is not limited to its application in DL; hence, the notation often uses $\mathcal{F}$ instead of $\mathcal{L}$ as a generic objective function.

2.2.2 Convolutional Neural Networks

Other than DNN, or Artificial Neural Networks[4] (ANN), Convolutional Neural Networks (CNN) [LeC+89] are designed to deal with grid-like input data like time series (one-dimensional) or images (two or three-dimensional) [GBC16, p. 326]. Although DNNs can also, in theory, process this form of data, in practice, computational complexity hinders effective results [ON15]. Consider the MNIST example again: The shape of the handwritten images in the MNIST dataset is $28 \times 28 \times 1$, as each pixel is presented as a neuron in the input layer; over 100 000 weights would connect it to a first hidden layer of 128 neurons. While this seems like a lot, most datasets depict higher resolution and colored images, reinforcing the need for a different approach.

[4] Artificial Neural Network is a more general term and includes NNs that are not considered to be deep due to the number of hidden layers.

Key to CNN is the *convolution* operation [GBC16, p. 326]. The operation performs an element-wise multiplication between a pooled area of the input and a *kernel* before summing up the resulting matrix. The kernel is a $n \times n$ matrix, whose values are changed during training.

The mathematical formula for this operation is written in Equation 2.8 [GBC16, p. 328]. Most ML libraries actually do not implement convolution but *cross-correlation* (Equation 2.9 [GBC16, p. 329]). The difference is that cross-correlation has no flipped kernel, whereas convolution only has it to keep its commutative property used in proofs [GBC16, p. 328f.].

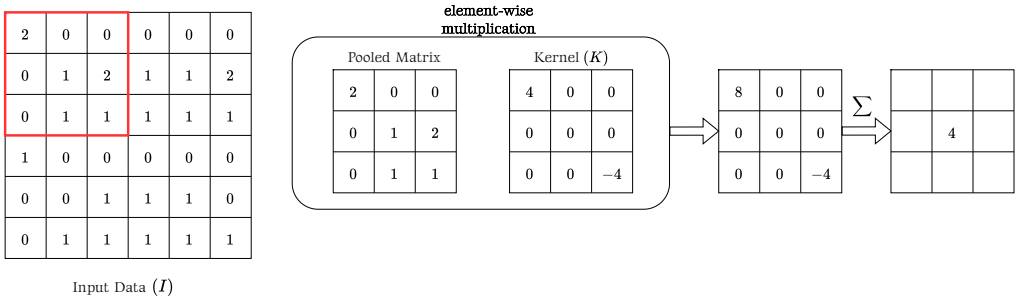


Figure 4 Convolution operation $(*)$ [ON15]

$$(K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n) \tag{2.8}$$

$$(K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{2.9}$$

By moving the kernel across the whole input data, the *activation map* is generated pixel by pixel. The step size of the kernel in each iteration is called the *stride* [LBH15]. Note how the activation map is smaller in size than the input data. Furthermore, pixel values close to the edge are actually used less than those in the center of the input data [Li+20b]. To overcome these drawbacks, CNNs use *zero-padding*. Essentially, extending the input data with edges of value 0 [Has19]. Underlining the necessity of zero-padding Goodfellow, Bengio, and Courville point out that "without zero padding, we are forced to choose between shrinking the spatial extent of the network rapidly and using small kernels–both scenarios that significantly limit the expressive power of the network" [GBC16, p. 343].

CNNs are composed of convolutional layers. Each convolutional layer is implemented by three stages. The first stage performs several convolutions, as just described. The second stage, analogous to ANNs, performs a nonlinear activation. Typically, the *rectified linear unit* (ReLU) function or modifications that mitigate the *dying relu*

[Lu+20] problem are used, as these have provided promising results [Aga19]. The third stage implements *pooling* [ON15].

Pooling. Pooling applies a summary statistic to a specific point of the output and its adjacent values [GBC16, p. 335]. Similar to the kernel in the convolution, the pooling layer will hover over the corresponding activation map. Take max-pooling as an example, as this is typically used within the pooling layer of the CNN [Chr+19]. Like the name suggests, the max-pooling layer will apply the MAX function [ON15]. Average-pooling likewise determines the average value across the pooling layer. Pooling's ability to reduce the spatial dimension is why it is also referred to as *downsampling*. This reduction is the main purpose of pooling, as it leads to less computational overhead [GK20; Zaf+22]. Additionally, pooling's ability to primarily maintain the most important features makes it less vulnerable to overfitting the training data [WG15].

The convolutional layers in a CNN are followed by fully connected layers, similar to those in an ANN. These layers are used to perform tasks such as calculating classification probabilities [ON15]. To make the input data processable for the fully connected layer, it is transformed into a one-dimensional vector in a *flatten* layer.

2.2.3 Neural Network Architectures

Most practical applications in the field of DL use existing NN architectures [Liu+19; LLZ19] rather than creating new ones from scratch, as these established architectures have shown state-of-the-art performances [RFB15; KSH12]. Other work may modify these familiar architectures to suit their use-cases better [Seo+20; SS19].

These widespread architectures were categorized by Minaee et al. for the domain of semantic segmentation [Min+20]. Encoder-decoder-based architectures are particularly popular for semantic segmentation [Min+20] and are also used in RNNs and Generative Adversarial Networks (GAN) [Goo+14].

Encoder-decoder networks are composed of two parts, namely the *encoder* and the *decoder*. The encoder's purpose is to produce a low resolution representation of the image fed to the net. "The problem lies on learning to decode or map those low-resolution images to pixel-wise predictions for segmentation. This part is named decoder and it is usually the divergence point in this kind of architectures" according to Garcia-Garcia et al. [Gar+17].

U-Net

The U-Net is such an encoder-decoder-based model. The U-Net architecture (see Figure 5) developed by Ronneberger, Fischer, and Brox is the most popular for

image segmentation and was originally designed for medical purposes. Its design allows for less computational power needed and the creation of many patches from a single data sample using data augmentation, resulting in less data needed to train the model [RFB15].

Since being introduced, U-Net has been used in many different domains beyond medical imaging. RS applications include the segmentation of buildings [ZLW18], sea-land [Sha+19], or roads [WM22].

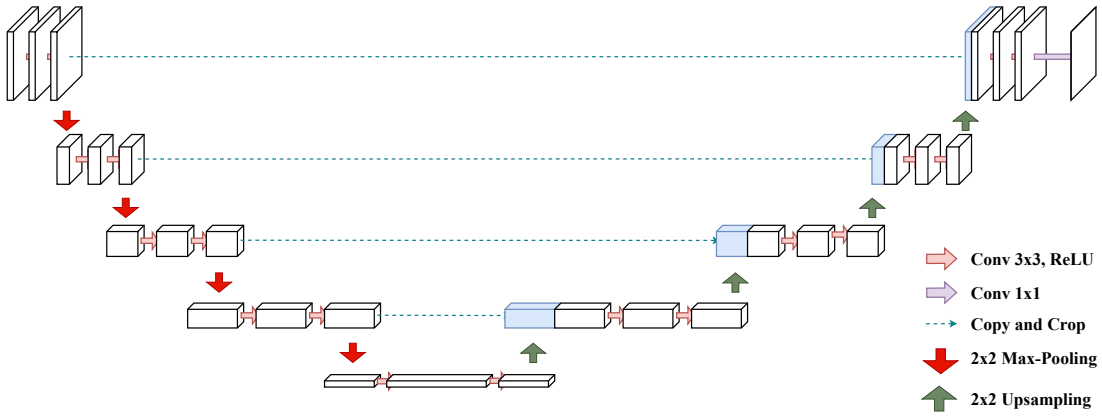


Figure 5 U-Net architecture with the encoder on the left, passing the bottleneck at the bottom, and decoding on the right [RFB15]

Both the encoder and decoder consist of four blocks, with each block having two convolution layers. The difference is that the encoder uses 2×2 max-pooling to downsample the image size while keeping the important features. The decoder upsamples the image in each block with a 2x2 convolution [RFB15]. Skip-connections between the encoder and decoder encourage keeping important features in the upsampling process [HT21].

Further architectural contributions are of such significance that they are used as building blocks in various models [Gar+17]. With encoder-decoder architecture, a general practice is to substitute the encoder with such building blocks; these substitutions are called the *backbone* [Du+20].

Residual Deep Neural Networks

Residual deep neural networks, or ResNets [He+15a], are often employed as a backbone. For this reason, a U-Net with a ResNet backbone has also been labeled *ResUNet* [Qi+20].

ResNets solve the problem of optimizing very deep NN. Optimizing deep NNs has been proven to be challenging, so in practice, they perform worse than shallow NNs

[He+15a]. The authors argued that deep NNs could perform at least as well as a corresponding shallow NN if the layers were copied and the remaining layers propagate the input to remain the same, essentially mapping the identity.

Figure 6 illustrates the basic building block, which forms the basis of ResNets, and shows how ResNets mitigate challenges regarding very deep networks. Given the identity x per skip-connection, only the residual $\mathcal{F}(x)$ is actually trained. Due to the initialization of weights, this part tends to be rather small, so only adjustments to x are learned in ResNets.

In contrast to ResNet architectures with a depth of 18 or 34 layers, deeper architectures (more than 50 layers) use bottleneck building blocks with a slightly different design [He+15a]. These target concerns pertain to computationally heavy training times with regard to the depth of the model.

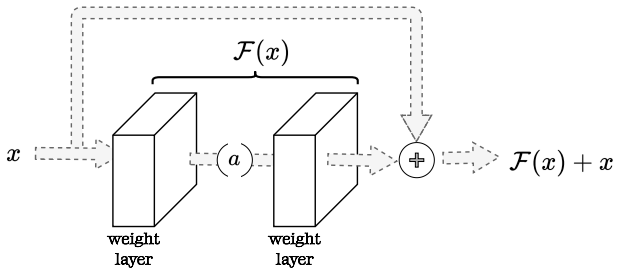


Figure 6 Architecture of ResNet Basic Building Block [He+15a]

2.2.4 Pre-Training or Random Initialization

The incorporation of existing architectures also allows for the adoption of weights pre-trained from models that have been previously trained on different datasets. This raises an important question: Pre-training[5] or random initialization?

Recent years have shown a trend towards the *pre-training and fine-tuning* paradigm [HGD18]. Instead of training a model from scratch, the model uses pretrained weights of large datasets. The intuition being that the model benefits from a jump start. Low-level features like edges are already "understood" by the model from the start. Converging to the higher-level task therefore becomes faster. Additionally, a lack of training data is compensated by the large amount of data that can be used for pre-training [HGD18].

Pre-training and fine-tuning with model weights achieved on the ImageNet [Den+09] dataset has become the standard for computer vision problems [Li+20a] due to state-of-the-art performance on object detection [Ren+16] or segmentation tasks [He+18; Che+17].

[5] Since the pre-trained weights were learned on a different dataset, literature also uses the term *transfer learning* [Ben12; WKW16].

Prior to the rise of *pre-training and fine-tuning*, models were trained from scratch. He, Girshick, and Dollár questioned the validity of the original approach given the new standard [HGD18].

Experiments show that random initialization is no worse than pre-training when training for long enough and utilizing normalization strategies [HGD18]. He, Girshick, and Dollár show that if the classification task is distinct from the pre-trained one, performance might even be inferior to training from scratch [HGD18; Ngi+18]. Contrary to this, Yosinski et al. show that transferring weights from distant tasks *does* lead to better results [Yos+14].

2.2.5 Hyperparameter Tuning

Hyperparameter tuning or search (more on that later) is about finding the appropriate set of hyperparameters λ in order to achieve the best training results.

More formal, given the learning algorithm $\mathcal{A}$ of the model $\mathcal{M}$, which takes in the training data $X^{(tr)}$ and a set of hyperparameters λ. The optimal hyperparameters λ^* (Equation 2.10 [CD15]) are set so that the objective function $\mathcal{F}$ performs best on the test data $X^{(te)}$ [CD15].

$$\lambda^* = \arg\min_{\lambda} \mathcal{F}(\lambda; \mathcal{A}, X^{(tr)}, X^{(te)}, \mathcal{L}) \tag{2.10}$$

Typically, the test data is reserved for final evaluation, while an additional dataset, the *validation set*, is used during the tuning process. The idea is to assess different hyperparameter configurations on the validation set. After identifying the best-performing configuration, its generalizability is confirmed using the test set [BB12].

While literature regarding pre-trained models describes the task as *hyperparameter tuning* [HGD18; Li+20a; Zop+20], literature outside the context of pre-training describes it as *hyperparameter search* [CD15]. Both refer to the same task. "The fine-tuning process is not different from learning from scratch, except for the weights' initialization" [Li+20a].

Popular options for hyperparameter tuning include grid search, bayesian algorithm, and genetic algorithm [AL21].

Grid Search. Grid search generates every possible combination across a predefined parameter space, thus creating a comparison between the cartesian product of hyperparameter settings. A drawback of grid search can be seen in the exponentially increasing possibilities when adding points within the parameter space. An

overly comprehensive grid search quickly leads to computationally infeasible runtimes [YS20].

Bayesian Algorithm. The bayesian algorithm is defined by an iterative process using two components: a surrogate model and an acquisition function. The surrogate model builds the distribution model of obtained results; the acquisition function uses this representation and decides for the next points to test in the parameter space to update the model again [YS20].

Genetic Algorithm. Genetic algorithm (GA) resembles the idea that fitter individuals are more likely to survive and pass on their genes to following generations. In hyperparameter optimization, each chromosome is represented by a hyperparameter. The algorithm initializes the first generation in which each individual is evaluated; better-performing individuals will then pass their genes to the following generation using mutation operations [YS20].

2.2.6 Regularization

Kukačka, Golkov, and Cremers define regularization as "any supplementary technique that aims at making the model generalize better, i.e. produce better results on the test set" [KGC17]. As NNs are trained using the training set, a general difficulty in ML is that the model does not only demonstrate good performance on the training set; it must especially demonstrate good performance on data it has not previously seen, which is resembled by the test set. The phenomenon of only performing well on the training data, but poorly on new data, is known as *overfitting* [Die95].

Early definitions for regularization were limited to regularization terms [Bis95]. These terms were added as a sort of penalization to the existing loss function [KGC17].

Take *weight decay* [KH91] as an example. Weight decay is the most popular regularization term [KGC17]. Equation 2.11 [KH91] shows the addition of weight decay to an error function E. As the model tries to minimize the loss function, the addition of $\sum_i \theta_i^2$ suppresses weights from growing too large, unless really necessary. Thus finding a small weight vector as a solution [KH91]. λ is a parameter determining the strength of the penalty.

$$\mathcal{L}_X(\theta) = E(\mathcal{M}_\theta(x_i), y_i) + \underbrace{\frac{1}{2}\lambda \sum_i \theta_i^2}_{\text{Weight Decay}} \tag{2.11}$$

With newer definitions, regularization is not limited to regularization terms but various aspects of DL. Equation 2.12, provided by Kukačka, Golkov, and Cremers, formalizes the loss function further to illustrate all the different categories for regularization [KGC17].

$$\arg\min_{\theta} \frac{1}{|X^{(tr)}|} \sum_{(x_i,y_i)\in X^{(tr)}} E(\mathcal{M}_\theta(x_i), y_i) + R(\dots) \tag{2.12}$$

$R(\dots)$ A classical regularization term independent of the target (see Equation 2.12)

$X^{(tr)}$ A modification of the training data itself (e.g., dropout [Sri+14] or batch normalization [IS15])

$E(\dots)$ An error function with regularizing effect (e.g., dice loss [Fid+18])

$\mathcal{M}_\theta$ The chosen model's network architecture (e.g., through pooling)

$\arg\min$ Optimizing procedures that are more likely to find flatter local minima (e.g., weight decay [KH91])

Note that regularizations are not mapped 1-1 to the different categories but can serve in multiple categories.

The Problem of Label Noise

The effect of overfitting is significantly more obstructive in the case of noisy data. Which leads to regularization playing a central role in the research of label noise [Liu+20; Son+22]. Label noise is one of the main challenges when working with real datasets [FV14; Son+22]. Label noise refers to any disturbance between the relationship of features and the corresponding class [Hic96].

Because of costly labeling [Han+21], real datasets show noise rates ranging from 8% to 35% [Son+22]. In addition, labeling can also be complex to the point where even domain experts are unable to label the data correctly [FV14]. Label noise can be further differentiated in feature noise and class noise. Feature noise, namely, describes a corruption in the features of a sample. Class noise is concerned with wrong labels [ZW04; FV14].

Memorization

In the case of noisy datasets, a phenomenon widely discovered is memorization. State-of-the-art models tend to be able to recall each training sample by memory. Experiments even proved this phenomenon on completely random data with no relation between features and target [Zha+17].

Memorization occurs only later in the training of a NN (see Figure 7). In earlier training, the model "focuses" on learning the easier patterns found in the data [Arp+17].

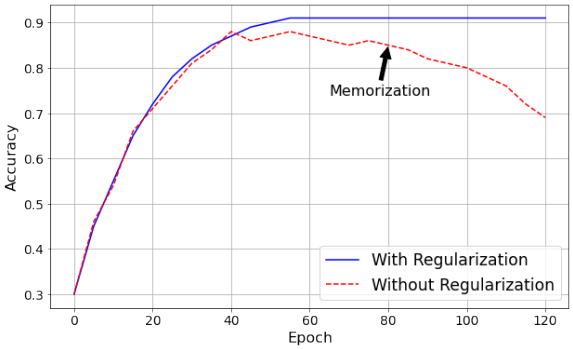


Figure 7 Memorization causing the model to overfit unseen data in later epochs as seen by the accuracy.

Early Learning Regularization

Early Learning Regularization (ELR) is a framework designed to address the challenges of learning with noisy labels, particularly to prevent memorization. ELR leverages the "early learning" phase, during which memorization has not yet taken place [Liu+20]. The underlying hypothesis is that during this "early learning" phase of training, the model is better able to discern the true label of noisy data.

In practice, ELR utilizes the model's early notion of correct labels using the target estimation t_i (Equation 2.13 [Liu+20]), a practice common in unsupervised learning [MCJ06], when no label is available. This approach is also used in semi-supervised learning [LA17], where only some labels are available.

The target vector keeps a moving average of the model's predicted probabilities for each sample i. These probabilities are outputs of the softmax ($\frac{\exp(\theta_i)}{\sum_j \exp(\theta_j)}$ [GBC16, p. 181]) or sigmoid[6] ($\frac{1}{1+\exp(x)}$ [LeC+12, p. 17]) for binary classification. The influence of more recent predictions of the model on the target vector is factorized by $\beta \in [0, 1]$. k resembles the k$^{\text{th}}$ training iteration.

$$t_i(k) = \beta t_i(k-1) + (1-\beta)\hat{p}_i(k) \tag{2.13}$$

$$\mathcal{L}_{\text{ELR}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{n}\sum_{i=1}^{N}\log(1 - \langle \hat{p}_i, t_i \rangle) \tag{2.14}$$

Equation 2.14 [Liu+20] shows the regularization term applied to the cross-entropy loss (Equation 2.1) in ELR. $\langle \hat{p}_i, t_i \rangle$ denotes the dot-product between the predicted probabilities $\hat{p}_i$ and the moving target vector t_i. λ is a factor that determines the weight of the regularization term. The logarithm is used to counter the exponentiality of the softmax function [Liu+20].

[6] In the field of AI, sigmoid serves as an alias for the logistic function (see `https://pytorch.org/docs/stable/generated/torch.nn.Sigmoid.html`). This notation is also preserved in this thesis.

The effect of the regularization becomes notable when looking at which circumstances trigger a high penalty. Consider a classification task with a target distribution for sample i $[0.2, 0.7, 0.1]$ in an early training iteration. This means early iterations generally had high confidence for *class two*. In the scenario, noise caused wrong annotations for the sample of *class one*, the real ground truth is indeed *class two*. Now let's consider a (i) first scenario in which the model's predictions tend towards the noisy label of *class one* and (ii) a scenario in which the model's predictions are mostly aligned with the target distribution of earlier iterations.

(i) The model predicts $[0.55, 0.35, 0.1]$ for the falsely labeled sample. ELR's term within the sum ($\log(1 - \langle \hat{p}_i, t_i \rangle)$) will subtract ~ 0.45 as the regularization term.

(ii) The model predictions are more aligned with the moving target, as it estimates $[0.3, 0.65, 0.05]$. The same term considered in (i) now results in ~ -0.73.

The second scenario (ii) shows how the "early learning" phase assumptions correct the model's loss for noisy labels when aligned with the target distribution, actively preventing memorization. Note that a negative loss is not a problem in model training. While typical regularization terms [KH91; Tib96] add a penalty to the existing loss, ELR compensates the loss by subtraction.

Early-Stopping

An implicit regularization strategy is *early stopping*. Early stopping is a technique used to automatically determine the optimal training time for DL models, with the goal of preventing overfitting by stopping training when the validation error no longer improves [GBC16, p. 243f.]. A more detailed description of the algorithm is described in Appendix A.1.

2.2.7 Evaluation Metrics

"Evaluation metrics play an important role in assessing the outcomes of segmentation models" [Jad20]. To overcome biases and misleadings of individual metrics, different metrics provide a more extensive view of the results. Typical evaluation metrics for binary segmentation are seen in Table 2, with TP referring to pixel correctly classifies as *true* and TN accordingly to correctly as *false* classified pixels. FP to pixel falsely classified as *true* or *false* in case of FN.

While accuracy (Equation 2.1) has a broad focus on positive and negative labels, IoU (equivalent to the Jaccard Index (JI)) (Equation 2.2) focuses on correctly predicting

Accuracy [CJ22]	$\frac{TP+TN}{TP+TN+FP+FN}$	(2.1)
Intersection over Union (IoU) [Cho24]	$\frac{TP}{TP+FP+FN}$	(2.2)
F1 [Sch+21]	$\frac{2 \cdot TP}{2 \cdot TP+FP+FN}$	(2.3)

Table 2 Evaluation Metrics

positive class predictions. The F1 score (Equation 2.3) looks similar to the IoU, but its purpose is to show the harmonic mean of precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+TN}$). Precision shows the ratio of true positive items to all positively predicted items. Recall, on the other hand, shows the relationship between true positives and all positives within the ground truth [Fer+18].

The area under curve (AUC) for the receiver operating characteristics (ROC) graph, short AUC-ROC, is one of the most popular evaluation metrics within the ML community [HKN19]. Instead of comparing the model's final prediction $\hat{y}$ it looks at how well $\hat{p}$ fits the labels on different decision boundaries. Measuring the true positive rate ($\frac{TP}{TP+FP}$) and false positive rate ($\frac{FN}{TN+FN}$) for different decision thresholds, the ROC curve is constituted, and the AUC for this curve depicts the model's performance as seen in Figure 8 [Faw04].

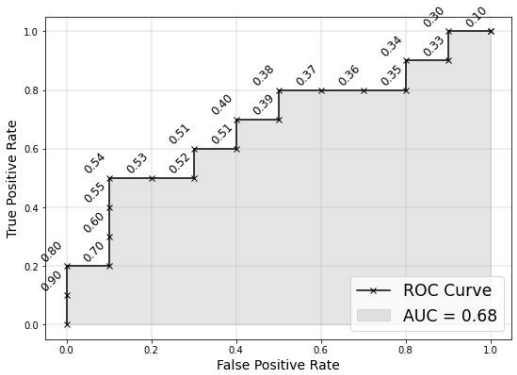


Figure 8 Exemplary ROC curve illustrating how AUC is calculated based on different decision thresholds

3 Data

The development of an updated forest mask is based on two main datasets. The first is satellite data obtained from the Sentinel-2 satellite, while the second is the *forestcover2000* dataset provided by Hansen et al. [Han+13].

(a) Sentinel-2 (b) Hansen et al. forest mask

Figure 9 The feature dataset (a) with multiple timestamps for the same ground truth (b)

Sentinel-2 Data

The Sentinel-2 mission, part of the Copernicus program, has two satellites orbiting the earth. Each point on the equator is visited with a frequency of 5 days, which enables to monitor changes in the Earth's surface. The data is organized in tiles, with each tile covering a surface area. Due to the frequency of data collection, each tile holds multiple timestamps (see Figure 9a). The data used in the thesis is from 2020 and has 12 bands ranging from colors (red, green, blue) with 10-meter resolution to near- and short-wave infrared of 20-meter resolution and 60-meter resolution coastal aerosol [GIS19]. Resulting in its input shape $\mathbb{R}^{R \times C \times H \times W}$ with C equal to 12 and R (revisits) ranging from 22 to 47. Since satellites are prone to errors, some of the data is faulty. Furthermore, in some cases, clouds hinder clear vision of the surface area and complicate the use of data analysis. The utilized dataset contains 104 832 samples with height and width of 128 pixels. The samples mostly covered Western Germany and parts of the Netherlands, Belgium and Luxembourg.

Hansen forest mask (forestcover2000)

The Hansen et al. forest mask depicts the relative canopy tree cover of each pixel, ranging from 0 to 100 (see Figure 9b). The forest mask has a resolution of 30 meters. The forest cover is from the year 2000, so any deviations from the satellite imagery will result in noisy data labels. It is available as input space $\mathbb{R}^{H \times W}$.

4 Methodology

This section introduces the approach followed to achieve the research goal. An illustration of the methodology is resembled in Figure 10.

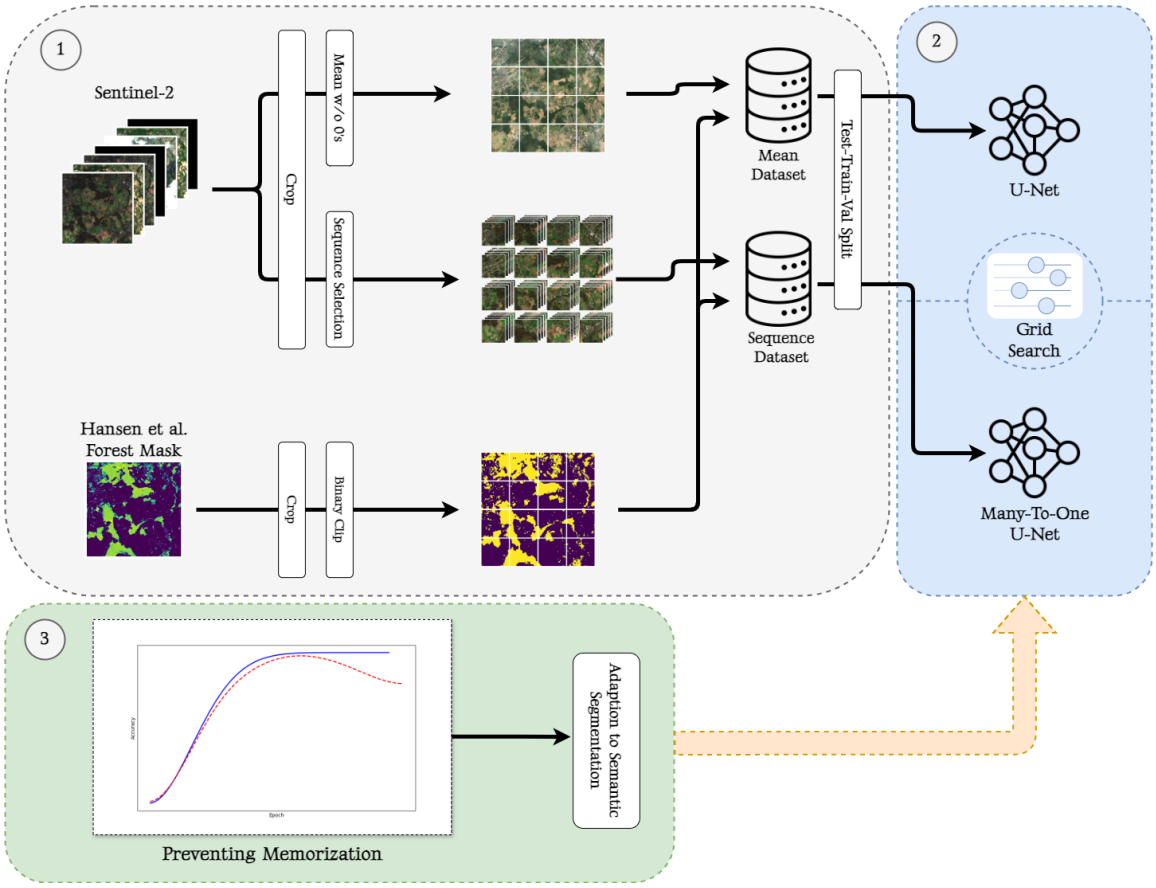


Figure 10 Illustration of the approach consisting of (1) data collection and preprocessing to build database objects; (2) training and tuning of two U-Net models, of which one has to be modified in order to use sequence input; and (3) introducing regularization to mitigate the effects of the noise within the data used.

4.1 Data Collection and Preprocessing

First, the necessary data has to be stored in a way that it is available to preprocess the data further. To ensure computational efficiency, the features as well as the ground truth are cropped. Two dataset objects will be implemented, the difference being their feature set. The *Mean Dataset* will take in the average values over every timestamp to create a single feature map. The *Sequence Dataset* will maintain the feature shape of different timestamps. The Hansen et al. label will be converted to consist of only binary values using a given threshold. This is necessary to classify the

images correctly in the semantic segmentation task. Lastly, both dataset objects will be divided into a training set, a validation set and a test set.

4.2 Model Development and Training

Using the *Mean Database*, a basic U-Net [RFB15] is first trained using the train set. Different hyperparameter settings are evaluated on the validation set using grid search. The best-performing model is trained with early stopping in place and used for the final results.

The *Sequence Database* is used to train a U-Net model modified to take in sequences as input. Therefore, a novel variant of the U-Net architecture will be introduced. Following the same procedure as the first model, after utilizing a grid search to find the best performing hyperparameters, early-stopping provides the model used for final results. Due to the similarities between the two models, a comparison will assist to evaluate the added value the sequential representation of the data will provide. The intuition being that the different timestamps will give the model a broader image of the "reality", which leads to more accurate segmentation.

4.3 Preventing Memorization

The time difference between the creation date of the Hansen et al. forest mask (2000), and the data observed by Sentinel-2 (2020) causes noise within the data. The resulting effect of memorization will be addressed using explicit regularization, preventing this effect. ELR will be implemented and tested to validate the regularization's effectiveness. As ELR has been developed for the classification task, modifying the approach is necessary to make it viable for semantic segmentation.

A final comparison can evaluate both the added value of the sequential satellite imagery and the added value of the regularization. As a result, the experiments will lead to a more optimal forest mask trained using more recent data with higher resolution than currently available forest masks.

5 Base Model

In this section, a simple model based on the U-Net architecture capable of masking forests is developed. The data is preprocessed, and experiments are conducted.

5.1 Data Preprocessing

The base model acts on aggregated data, which allows for easier use later in the training process. This aggregation is part of the preprocessing.

Famili et al. clearly divide the preprocessing into two subprocesses: solving problems with the data and preparing the data for data analysis [Fam+97].

Step 1: Solving problems with the data

The inaccuracy of satellite data must be addressed. An average over the timestamps of a single sample without addressing pixel values of zero overcomes errors in the data, as well as mitigating the noise caused by clouds. This aggregation is shown in Figure 11. Note how the clouds of the original images are slightly visible in the top left.

Figure 11 Mean over multiple timestamps ignoring values of zero within the features

Another option is to take the median without zeros and thus mitigate more of the outliers. Intuitively, this would be more sensitive to the high values created by clouds; however, experiments showed comparable results as to the mean. Furthermore, the median calculation was computationally more intensive.

In a next step, the values are normalized to the ranges 0-1 for computationally efficient processing with CUDA. Min-max normalization is a popular method that preserves the relationship between the original data [HKP11].

$$\hat{T}_c = \frac{T_c - \min(T_c)}{\max(T_c) - \min(T_c)} \tag{5.1}$$

Equation 5.1 [PS15] shows the min-max normalization over a channel c, with T being a sample's tensor and $\hat{T}$ the normalized tensor.

For the forest mask, a threshold is needed to provide binary targets for the segmentation task. After manually looking up different thresholds, a value of 20% tree cover was chosen.

Step 2: Preparing for Data analysis

The average of the satellite data samples were stored offline, resulting in an 8x increase in training speed in comparison to aggregating the images on loading. Moreover, both features and labels were cropped to $H = 128$ and $W = 128$, to allow for feasible batch sizes.

5.2 Experiments

A U-Net implementation was imported from the *pytorch segmentation models*[7] package as the base model. This implementation allows for ResNets as encoders of the U-Net. In addition, the model can be initialized using pre-trained weights established on the ImageNet dataset [Den+09].

5.2.1 Model Comparison

Prior to the search or tuning of a set of hyperparameters, it is first necessary to determine which model architecture to utilize and whether pre-trained weights should be used. To evaluate this decision, each of the most prominent ResNet encoders [WTJ21] was trained using the preprocessed dataset (see Section 3).

	ResNet34	**ResNet50**	**ResNet101**
pre-trained	**87.24**	86.74	87.20
random-init	86.09	85.83	86.10

Table 3 Validation Accuracy (%) after 15 training epochs using different encoder models as well as weight initializations. The achieved validation accuracies are not comparable to the final ones due to use of uncropped data.

Using *ResNet34* with already pretrained weights on the ImageNet dataset [Den+09] yields the best results (Table 3), which leads to further experiments using this combination.

[7] `https://smp.readthedocs.io/en/latest/models.html`

The finding of pretrained models to establish higher accuracy, especially with this short training time (*epochs* = 15), is aligned with findings by He, Girshick, and Dollár. "Pre-training has enabled state-of-the-art results on many tasks, including object detection, image segmentation, and action recognition" [HGD18].

5.2.2 Hyperparameter Tuning

In the following experiments, grid search has been used to evaluate the parameter space due to its simple application both conceptually and technically. Furthermore, Alibrahim and Ludwig provide comparable results across the different techniques, with only GA having slightly superior results [AL21].

In a first run, different optimizers were evaluated using the previously selected model on the preprocessed datasets. Because of computationally limited resources, only regular mini-batch gradient descent, RMSProp [HT12] and Adam [KB17] have been considered on the learning rate set $\eta = \{0.1, 0.01, 0.001\}$.

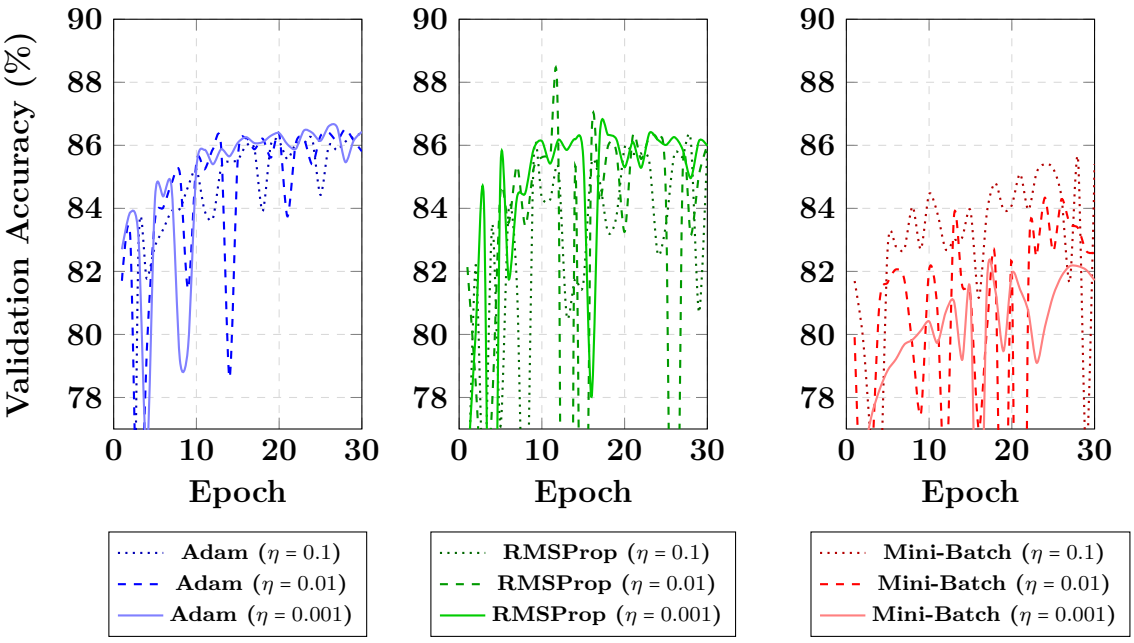


Figure 12 Comparison of Optimizers and Learning Rates on Validation Accuracy (%) over 30 training epochs

Figure 12 shows the optimizers' validation accuracy with a batch size of 16. Adam proves to be the best-performing among the three optimizers, as well as the most stable. Regular gradient descent shows the worst accuracy on the validation set, as it only crosses 85% accuracy in two iterations with $\eta = 0.1$. This, however, might be *lucky shots* attributed to the high fluctuation. While also presenting high variance, RMSProp's validation accuracy is well competing with Adam.

6 Many-To-One U-Net

This section introduces the *Many-To-One U-Net* (M2O U-Net). A modification of the U-Net [RFB15] capable of using sequential data as it's input to gather higher qualitative information for segmentation than with an aggregated representation of the input data.

6.1 Fusion Block

A Fusion Block fundamentally addresses the challenge of effectively reducing the dimensionality of sequential data for further passing the data through the U-Net. Such Fusion Block has been introduced by Cornebise, Oršolić, and Kalaitzis [COK22] and was also adapted by Wolters, Bastani, and Kembhavi [WBK23]. Both used the sequential data format for scaling the original images' resolution in the field of *Super-Resolution*.

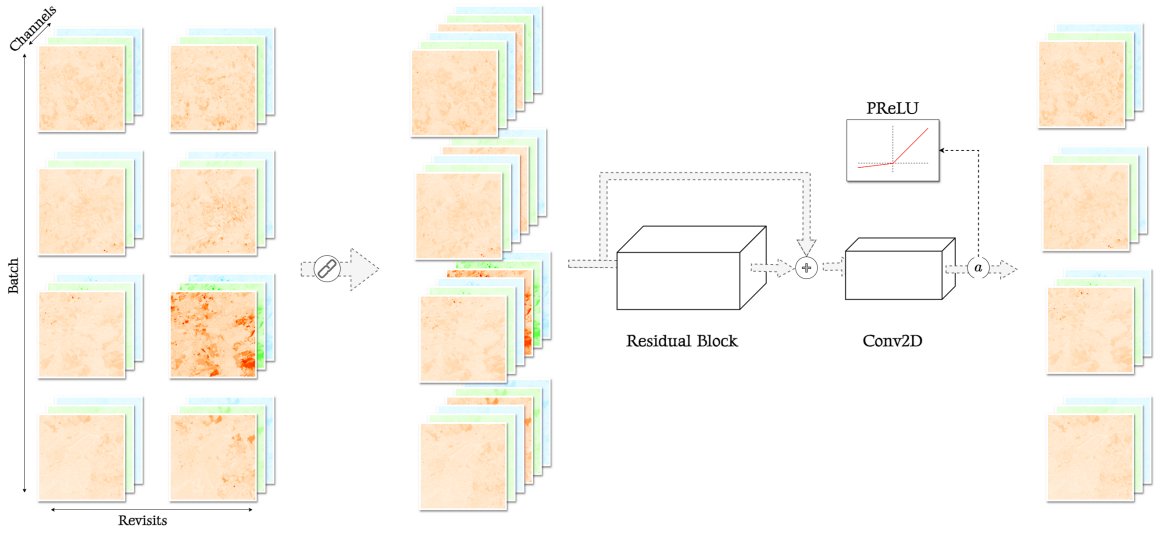


Figure 13 Structure of the Fusion Block, which reshapes the given input along the channel dimension and outputs the data's original form, except for halving the dimension of revisits. The representation of only three channels is for illustration purposes only; in practice, all channels are used.

A propagation through the Fusion Block proceeds as follows:

1. **Input Concatenation:** The input sequences are first divided into two parts of equal length across the dimension of revisits. The second part is then concatenated to the first across the channel dimension.

2. **Convolutional Layer:** The concatenated data is processed through a residual block [He+16] and a 2D convolutional layer, which captures in-

tricate features and reduces the channel dimension back to the original. Afterward, the PReLU activation function [He+15b] transforms the resulting feature maps.

3. **Final Reshape:** Finally, the data is reshaped to its original shape, with revisits being halved ($\mathbb{R}^{B\times R//2\times C\times H\times W}$).

The halving of revisits in each iteration creates the need for multiple propagations in case more than two revisits are fused into a scalar representation. Ideally, revisit length as powers of two (meaning $\log_2(x) \in \mathbb{Q}$) are used. Otherwise, black feature samples will be used as padding to reach a sufficient number of revisits.

6.2 Model Architecture

In the M2O U-Net, a sequence of Fusion Blocks is placed before passing output data through the encoder of the U-Net. In comparison to the base model (see Section 5), the best representation to use for the U-Net is learned instead of given as an aggregated sample. The data is propagated through a Fusion Block iteratively until the revisits dimension is fully consolidated.

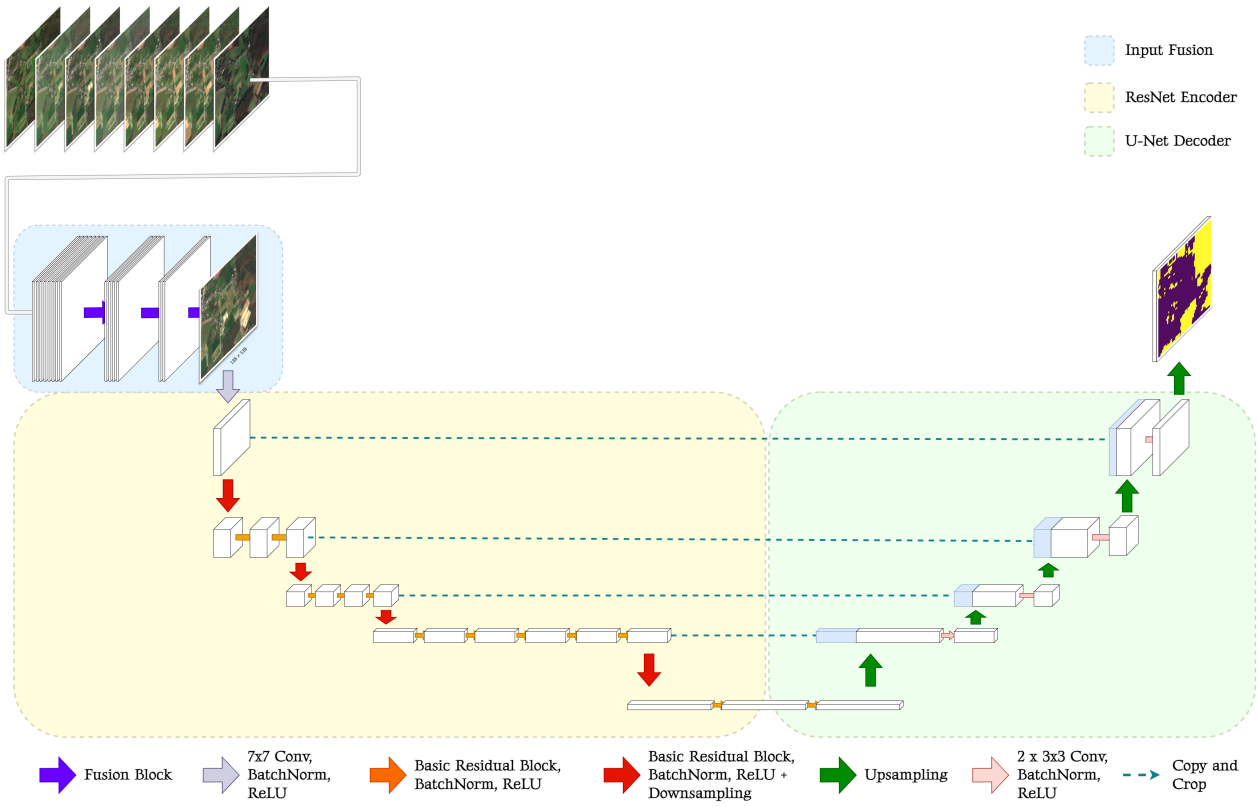


Figure 14 Architecture of the M2O U-Net consisting of a Fusion part, the ResNet encoder, and the decoder as originally found in Ronneberger, Fischer, and Brox's U-Net.

Figure 14 demonstrates the architecture of the M2O U-Net. The input sequence is first fused in the Fusion part. For the example of 8 revisits 3 Fusion Blocks are necessary to provide the encoder with the correct input shape reduced in the revisits dimension ($\mathbb{R}^{B\times C\times H\times W}$). This learnable representation of the input data is encoded using a ResNet [He+15b]. Due to the results in Section 5, here ResNet34 is employed as the encoder. The decoder of the M2O U-Net was kept the same as in the original U-Net [RFB15].

To ensure comparability with the base model, the modification is integrated with the *pytorch segmentation models* U-Net design. The *ResNet34* encoder structure is therefore aligned with the base model, as well as the usage of pre-trained weights.

6.3 Data Preprocessing

Following the procedure in Section 5.1, the preprocessing is divided in Famili et al.'s two steps [Fam+97].

Step 1: Solving problems with the data

Problems with the data constitute (i) the difference in sequence length, which should be equally arranged for the model, and (ii) noise, which has been addressed before.

Both problems are addressed using the *Sequence Selection* procedure (see Algorithm 1).

Algorithm 1 Sequence Selection

Input: x ▷ List of revisits
Input: *seq_length* ▷ Desired length of the sequence
Output: *sequence* ▷ Resulting sequence
$non\_black\_samples \leftarrow [sample \text{ for } sample \in x \text{ if } sample \neq \mathbf{0}]$
$non\_black\_samples \leftarrow \text{sort}(non\_black\_samples, \text{ key} = \max)$
$sequence \leftarrow non\_black\_samples[: seq\_length]$
if $\text{len}(sequence) < seq\_length$ **then**
    $num\_to\_duplicate \leftarrow seq\_length - \text{len}(sequence)$
    $sequence \leftarrow \text{append}(sequence, \text{random.choice}(sequence, num\_to\_duplicate))$
return *sequence*

The algorithm selects a sequence length of revisits with the lowest max values. This is done because cloud coverage results in very high pixel values; selecting the revisits with the lowest maximum most likely selects revisits that are not prone to showing cloud coverage. In case the demanded sequence length surpasses the available non black image data, random samples are duplicated to ensure a consistent data format

across the dataset. Normalizing the input data as well as converting the targets to binary format is done analogously to the base model preprocessing (see Section 5.1).

Step 2: Preparing for Data analysis

Similarly, the data has been cropped and stored offline. Note that cropping has been done in advance of Sequence Selection. Therefore, selecting error-free input data is even more likely.

6.4 Experiments

In the following, multiple experiments using the M2O U-Net are conducted.

6.4.1 Optimal Sequence Length

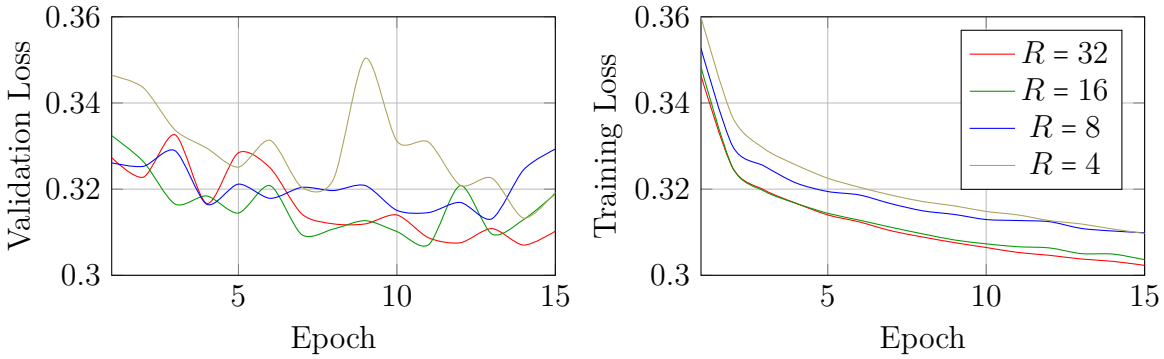


Figure 15 Loss comparison for different sequence lengths

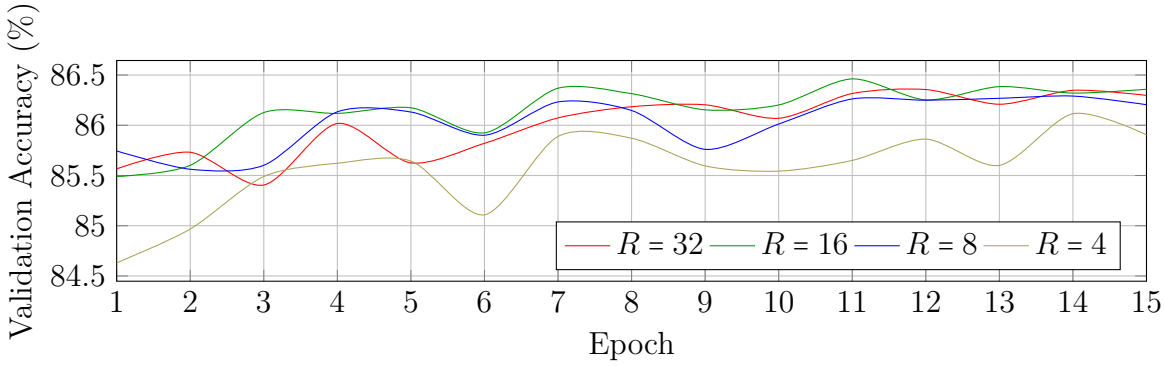


Figure 16 Validation Accuracy (%) for different sequence lengths

Despite the implementation using a selective procedure to determine the input data, the question "*Is more data generally better?*" arose. Especially considering the data's noisy nature. To answer the question, multiple training iterations were performed using sequence lengths of $4, 8, 16$, and 32 over 15 epochs to determine the optimal sequence length.

Figure 15 shows how more revisits in the feature set generally reduce the loss. However, the degree of change from 16 to 32 revisits is minimal, showing that the explicit selection of revisits that are more likely to be error-free is effective.

The validation accuracy depicted in Figure 16 confirms this behavior. A sequence length of 4 provides the worst accuracy on the validation set, while 8 revisits show superior accuracy, and 16 revisits prove to be even better. Doubling the information from 16 revisit sequences ($R = 32$) does not enhance the performance. Consequently, a sequence length of 16 revisits is chosen for further experiments. Although more data can be beneficial, it also incurs increased computation time due to the mandated use of multiple fuzes.

6.4.2 Hyperparameter Tuning

The extent of hyperparameter tuning was kept the same as with the base model. This means measuring Adam, RMSProp, and regular mini-batch gradient descent on the dataset. The covered parameter space is limited to $\eta = 0.001$. This is due to a diverging performance observed using learning rates higher than 0.001. Figure 17 shows the results of the hyperparameter tuning process. Again, Adam [KB17] performs the best, closely followed by RMSProp [HT12], except for an unusual drop in the 5^{th} epoch. As a result, the Adam optimizer is used in further experiments.

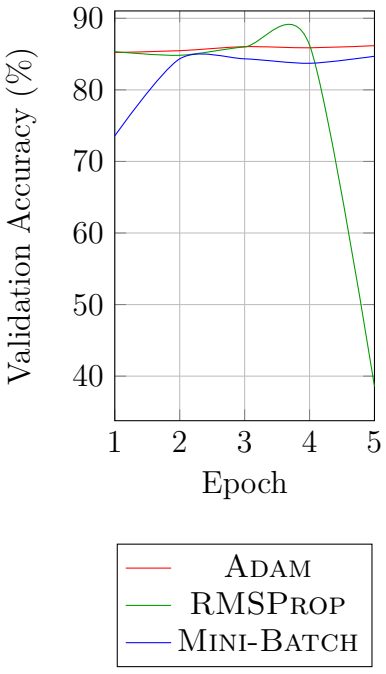


Figure 17 Using different optimizers with the M2O U-Net over 5 epochs with learning rate $\eta = 0.001$

6.4.3 Alternative Attempts

Prior to the development of the M2O U-Net, experiments with a different model architecture were conducted. In this architecture, instead of using the Fusion Block before propagation through the encoder, integrated the Fusion Block at the skip-connections between the encoder and decoder of the U-Net. Various training iterations have shown that this architecture cannot compete with the original U-Net using aggregated samples, as showcased in Section 5. These experiments lead to the assumption that utilizing the U-Net with sequential data is not easily done. A comprehensive review of the architecture's comparison to the base model can be found in Appendix A.2.

7 Challenges in Model Training

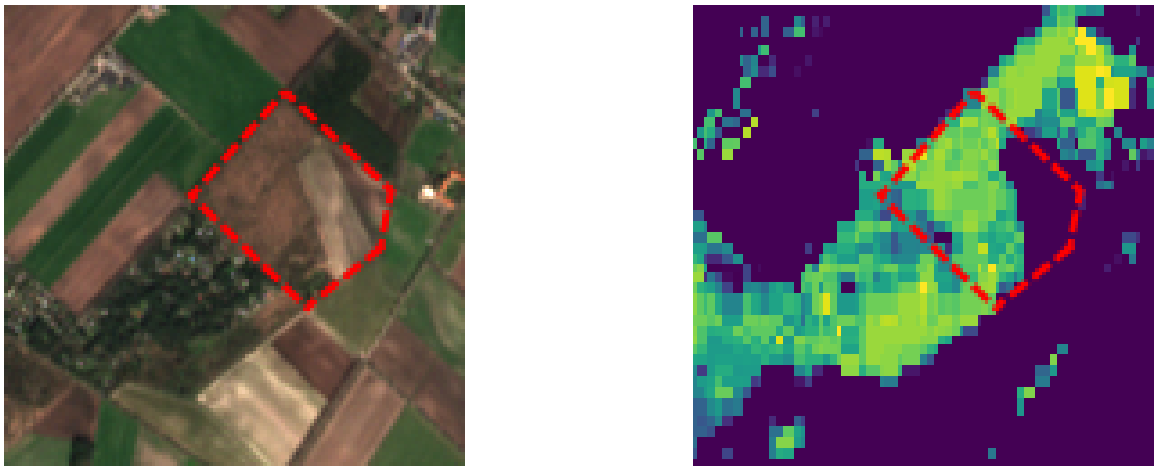
This section explores noise within the provided dataset and experiments with means to prevent the noise from impairing the model's performance.

7.1 Noisy Data

For the forest segmentation task, noisy data constitutes a main challenge. Noisy data is (i) observed in feature noise seen in cloud cover and (ii) in the outdated data resulting in label noise.

(i) Cloud cover prevents the satellite from observing the ground truth of the Earth's surface. Furthermore, cloud shadows result in further feature noise. The challenge of cloud cover is well-known in RS [PTH11], and many approaches to remove cloud effects on satellite images have even been developed [San+20].

(ii) The outdatedness of label data causes ground truths that falsely label forest cover where there is none on actual data and vice versa. Figure 18 demonstrates an example of feature noise resulting in positive labels for *forest*, where the labels should be negative (*no forest*).



(a) Sentinel features from 2020

(b) Hansen et al. label from 2000

Figure 18 Feature noise caused by part of a forest that has been deforested since 2000.

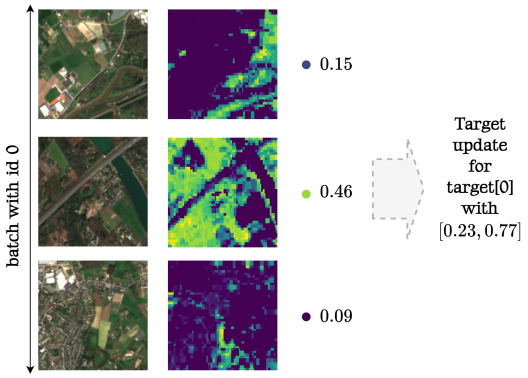
7.2 Early Learning Regularization for Semantic Segmentation

The presence of noise within the data results in the model memorizing these false labels later in training. To mitigate the effects of noisy data, Early Learning Regularization has been applied to both the base model and the M2O U-Net model.

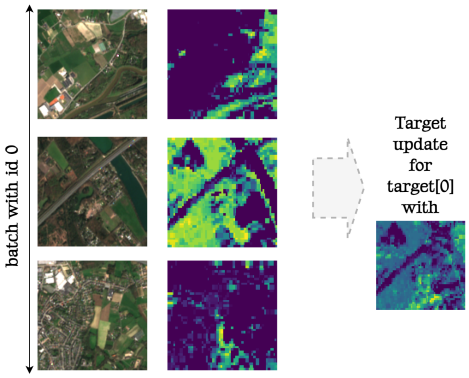
Since ELR is originally applied to classification tasks, modifying the loss calculation was required. First, because of the difference in classification and segmentation. Second, because of the case of binary classification instead of multi-classification.

The proposed regularization by Liu et al. uses the softmax class predictions to build the target estimate for each class [Liu+20] (see Figure 19). With segmentation, such predictions are available for each pixel across the samples in the batch. Therefore, the target estimation comes with two options for the scope. The target could (i) be based on the overall class distribution per batch ($t \in \mathbb{R}^B$) or (ii) even further be differentiated to evaluate the target across each pixel position for the batch ($t \in \mathbb{R}^{B \times H \times W}$). Figure 20 illustrates both approaches.

Figure 19 Target estimation in ELR for classification [Liu+20].



(a) Approach 1: Calculating the target using the feature mean predictions.



(b) Approach 2: Calculating the target using the feature's means further differentiated per pixel location.

Figure 20 Two different target estimation approaches for semantic segmentation.

Because of the binary ground truth, CE (Equation 2.1) has been switched with BCE (Equation 2.2). Furthermore, the counterprobability has been explicitly included in

the calculation of the regularization term. A code extract showing the implementation of ELR for semantic segmentation with binary labels using the batch mean can be seen in Appendix A.3.

The comparisons between the different approaches (Figure 21 & Figure 22) illustrate, except for some small variations, that there is no considerable difference between adapting ELR on a more general batch dimension or also differentiating the target on the pixel locations.

The difference in validation loss can be partly explained by random circumstances in the training of neural networks. Which is supported by the fact that the training loss shows less variance.

The validation accuracy paints a similar picture. Although the 3rd and 5th epochs show a temporal drop in performance for both approaches, the general performance is mostly equal. Due to less computational overhead, the *on batch* approach is kept for the final results.

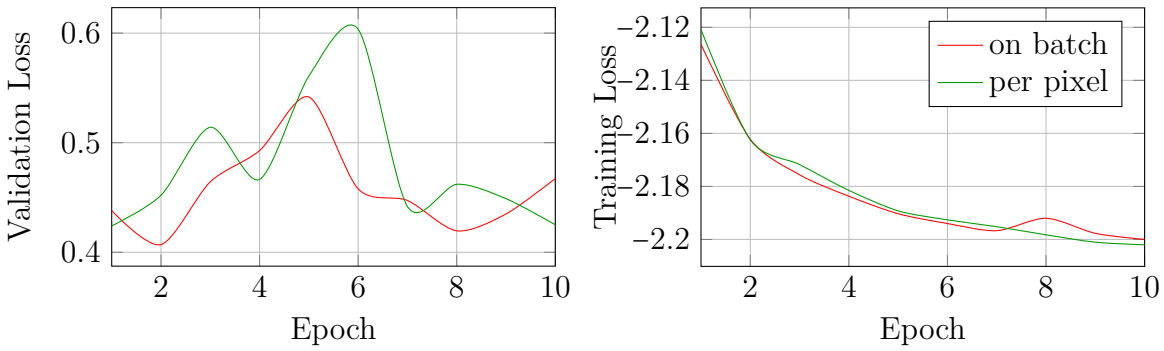


Figure 21 Loss comparison of two ELR approaches suitable for semantic segmentation over 20 epochs

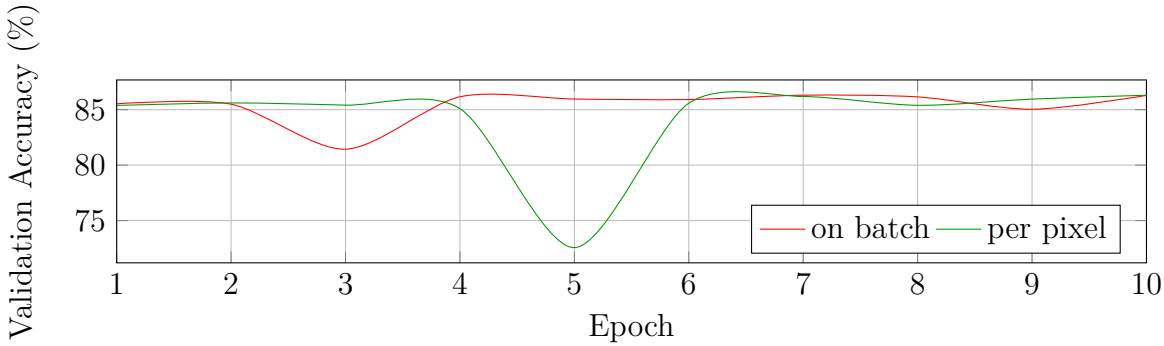


Figure 22 Validation Accuracy (%) comparison of two ELR approaches suitable for semantic segmentation over 20 epochs

Other regularizations related to ELR, like Co-Teaching [Han+18] or MentorNet [Jia+18], were found not to be applicable to semantic segmentation at their base.

7.3 Prevention of Memorization

In this section, the prevention of memorization during model training is analyzed by comparing BCE and ELR using the base model over a long training period.

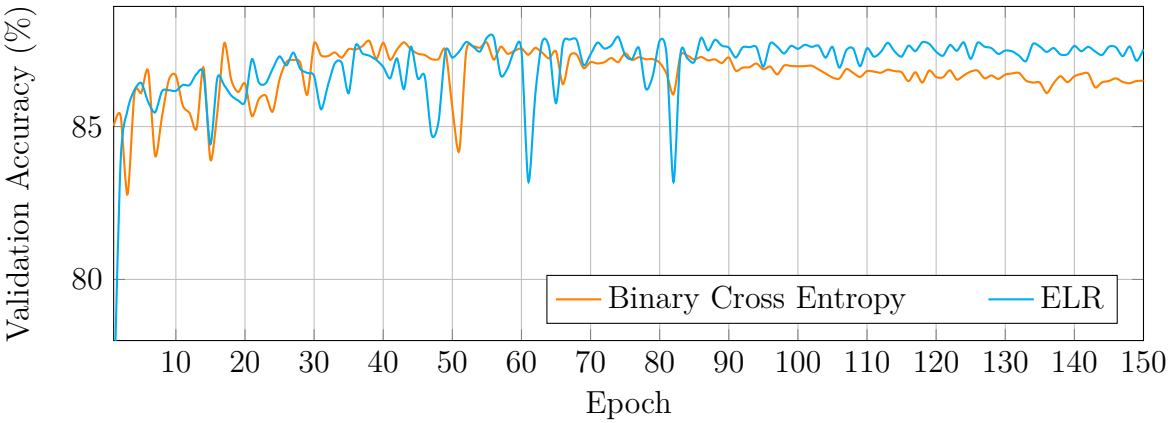


Figure 23 Validation Accuracy comparison for Binary Cross Entropy and ELR over 150 epochs

The experiment (Figure 23) is conducted over 150 epochs to provide a comprehensive view of the long-term performance of each method. The plot indicates that the validation accuracy for BCE declines significantly after approximately 90 epochs. This decline suggests that the model begins to overfit, memorizing the training data rather than generalizing from it. ELR, on the other hand, maintains a more consistent validation accuracy throughout the 150 epochs. This consistency implies that ELR is effective in mitigating the effects of memorization. The results highlight that while BCE is the standard loss function and does not inherently aim to prevent memorization, ELR demonstrates a clear advantage in sustaining higher validation accuracy over a long training period.

8 Overall Comparison

This section provides a final comparison between the M2O U-Net (Section 6) and the base model (Section 5). Furthermore, the added value of using the adapted ELR (Section 7.2) is examined.

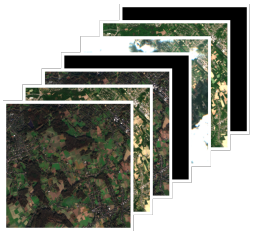
Model	Metric	BCE	ELR
 **Base Model**	Accuracy (%)	86.111 ± 0.790	**86.441** ± 0.013
	IoU	**0.498** ± 0.013	0.492 ± 0.006
	F1	0.802 ± 0.001	**0.804** ± 0.002
	AUC-ROC	**0.930** ± 0.002	0.907 ± 0.06
 **Many-To-One U-Net**	Accuracy (%)	86.218 ± 0.300	85.970 ± 0.458
	IoU	0.469 ± 0.040	0.487 ± 0.017
	F1	0.793 ± 0.014	0.799 ± 0.003
	AUC-ROC	0.929 ± 0.001	0.912 ± 0.003

Table 4 Average test evaluation metrics for the four different combinations using ELR and BCE on both the base model and M2O U-Net with early-stopping (*patience* = 5, *min_delta* = 0.01). Three training runs were taken with each model. The training runs took 14–28 epochs, lasting between roughly 4.5 and 9.3 hours. One run, which only trained 11 epochs, got filtered out because of assumptions that the run stopped too early due to the early-stopping configuration.

The results (Table 4) provide a mixed picture. While ELR improves the accuracy of the base model, it does not in the case of the M2O U-Net.

The base model generally performs better in terms of IoU and F1 score, whereas the M2O U-Net shows a slight advantage in accuracy with BCE and AUC-ROC with ELR. These differences can not be attributed to the architecture, but rather to the variance in the results. Overall, the base model is slightly superior to the M2O U-Net in consideration of the metrics used.

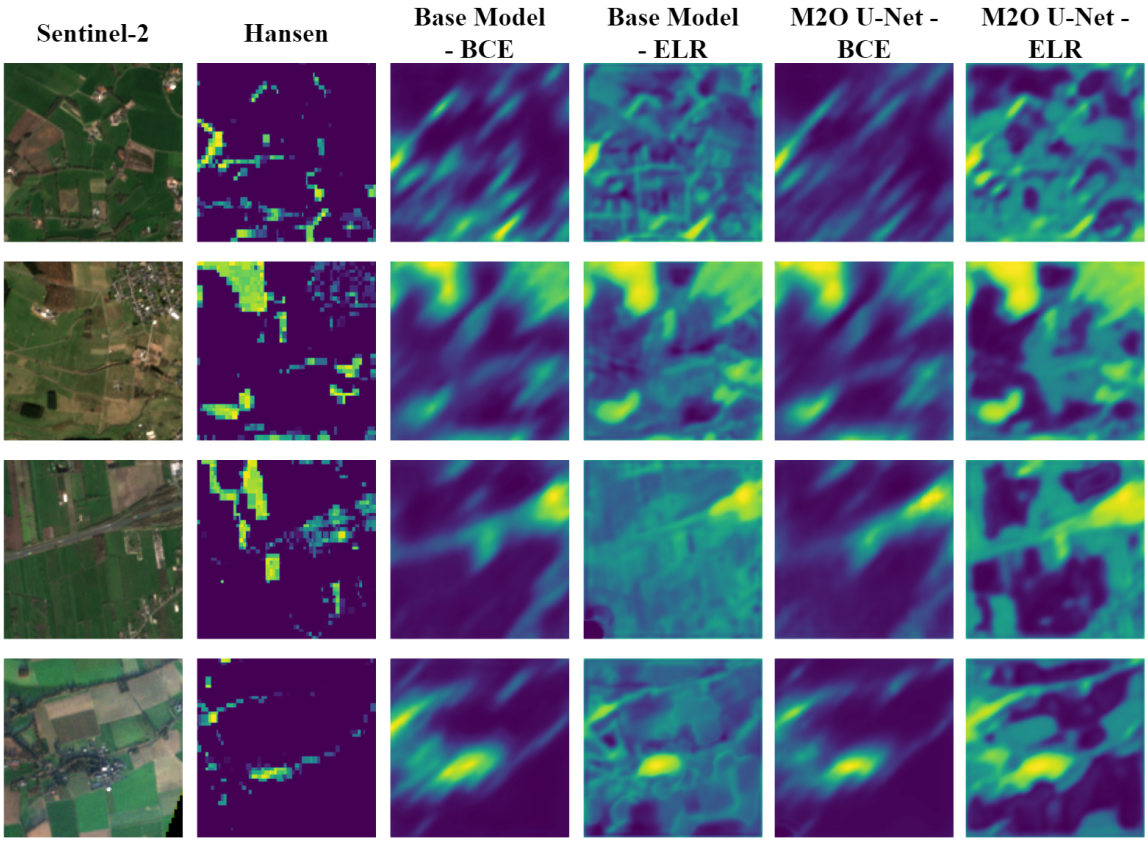


Figure 24 Comparison between each model's prediction for randomly selected samples within the test set

Figure 24 gives deeper insight into the prediction of each model. Note that while Hansen et al. depict the relative tree cover on a scale of 0–100, the predictions estimate the probability of forest, with $\hat{p} > 0.5$ leading to a positive estimation. Therefore, the values of Hansen et al. and the models, here outlined by the intensity of a pixel, should not be directly compared, but rather the general assumption between ground truth and prediction.

The most notable difference is in the comparison of predictions made with BCE and ELR. Predictions with ELR are less faded and more clearly differentiate a structural outline. This may be because it relies less on the ground truth and more on the model's early intuition.

The difference between the base model and the M2O U-Net is only visible with ELR. The M2O U-Net has clearer predictions regarding the negative pixel classification, while the base model seems to be less decisive for the pixels, especially visible in the third image. In the upper left of the third image, label noise is depicted in the ground truth. All models correctly estimated the region to have *no forest*.

9 Discussion

In this section, I present a detailed analysis of the findings of this thesis and place them within the context of the existing literature, highlighting their significance and relevance. Furthermore, I address the limitations encountered during the research process and discuss how these limitations may have influenced the results. Lastly, I provide clear recommendations for future research directions.

9.1 Implications of Findings

No Clear Recommendation. Although the main approaches (base model & M2O U-Net) significantly differ in the input data shape and the way the data is handled, a clear recommendation based on the comparisons is not establishable. The results show similar performance with a slight favor for the base model; a clear pattern is not depicted. I suggest further research to evaluate the M2O U-Net based on different datasets that (i) have less noise within the data, and (ii) have various sequence lengths available for experimentation. This way, more external factors are eliminated from the equation, targeting the effect of *actively training* a collective representation instead of basic aggregation.

ELR vs. Early Stopping. The evaluation metrics show no apparent difference between the use of BCE and ELR. One question that arises from the results is: *What is the benefit of ELR when early stopping is used?* With early stopping, the model is less prone to memorizing the noisy data, but ELR is designed to do just that. It may be that ELR is beneficial because the models are on the verge of memorization when early stopping comes into effect, but this would need to be verified. The prediction maps (Figure 24) give an idea of an existing benefit even with early stopping, this illustrated difference needs more exploration for clearer insights.

Integration of Sequential Data in U-Net. My findings underline the certain complexity constituted in integrating sequential data in the U-Net architecture. While existing research has already incorporated sequential data with U-Net [COK22; AR19], my experiments suggest that a similar integration yields undesirable results (Appendix A.2). The M2O U-Net, while using sequential data, does not really interfere with the U-Net's original logic. Attempts at incorporating recurrent logic with long short-term memory (LSTM) [HS97] were unsuccessful and therefore not covered in the thesis. I consider LSTM to be more applicable to the case of continuing a time series, which was not the case here. I suggest further research to explore the causes for worse performance with the model used in Appendix A.2.

9.2 Limitations

Limited Hyperparameter Tuning. The hyperparameter-space was very limited. Only three optimizers have been analyzed: Regular mini-batch gradient descent, RMSProp and Adam using three learning rates $\eta \in \{0.1, 0.01, 0.001\}$. Two factors mainly contributed to this decision, which interact in a reinforcing feedback loop. (i) The computation time needed for a single experiment (further explained in the next point) and (ii) that for comparison purposes the base model and M2O U-Net should conduct the same hypertuning process, meaning each expansion of the parameter space increases the computation time twofold. I urge further research to increase hyperparameter search, especially for ELR, which was basically left out of hyperparameter search in this thesis. This could help to explore the real added value of the regularization more.

Extensive Computation Times. The models' architecture using a ResNet encoder demands longer computational processing. Nevertheless, the most significant factor is the data itself. The Sentinel-2 data shape, including 12 channels and multiple revisits, is very extensive. Although preprocessing the data could reduce the computational time during training significantly, the process of taking the data, transforming it, and storing it could also take up to 12 hours. The reduced time varies depending on what data format was used, but also took between 5-9.5 hours approximately and depending on the epochs used. Such expensive computations forced me to be very decisive with the experiments conducted.

Especially with ELR, memorization could not have been explored enough. A definite representation of the memorization effect is expressed with long training times. Additionally, training runs without ELR have to run just as long in order to provide comparability. Apart from one experiment showing a mitigating effect for memorization, more experiments could give deeper insight into how to mitigate memorization further. Such experiments were too computationally expensive.

Dependency on Hansen et al. Label. The model learns to predict forest segmentation based on the Hansen et al. [Han+13] target. While I am aware of the noise within the data, which I mainly attribute to the creation date of the forest mask, I generally assume the target to be correct. Different sources have expressed concerns with Hansen et al. forest mask due to (i) its *forest* definition and (ii) classification errors.

(i) With Hansen et al., forest is defined as "all vegetation taller than 5-meter in height", not following the Regulation on Deforestation Free

Products[8] (EUDR) definition of forests. This is problematic as it also includes monocultures of oil palms, rubber, or eucalyptus, which pose threats to tropical biodiversity [Tro+14].

(ii) Even crops that are outside the definition, like rice or sugarcane, are classified as forests. According to Cunningham, Cunningham, and Fagan all row crops show an error of being classified with more than or equal to 89% tree cover [CCF19]. While classification errors have been revealed mostly in tropical areas, it still raises questions about the quality of the mask.

9.3 Future Directions

Some points were already mentioned in *Implications of Findings* (Section 9.1), namely putting the M2O U-Net to experiment on other datasets with less data noise, comparing ELR's effect for classification and segmentation and diagnosing error causes as depicted with the alternative approach.

Subsequently, various existing architectures can be applied to the preprocessed dataset. While I intentionally based the forest mask on the U-Net architecture, considering its proven effectiveness and existing results [Sha+19; Liu+19], other segmentation models might offer superior performance, especially given the sequential nature of the data. Evaluating similar models, such as V-Net [MNA16] or recurrent models, could provide deeper insights and potentially better results.

On top of that, I encourage researchers to further investigate the memorization effect and ELR's ability to mitigate it. As early stopping supposedly stops updating the model weights at a point where generalization is achieved, it is unclear whether ELR provides a benefit in that case. The memorization effect, which is handled by ELR loss, comes into effect only later. Subject to future research might be to evaluate ELR and BCE during a learning phase not visually affected by memorization.

[8] `https://green-business.ec.europa.eu/deforestation-regulation-implementation_en`

10 Conclusion

The research goal of *developing an updated forest mask for Germany* of 10-meter resolution based on Sentinel-2 data can be seen as fulfilled. A model capable of processing satellite data in common format to mask forests has been developed. Even though the experiments cover mostly smaller scale segmentations, a higher scale forest mask can equally be established without difficulty (see Appendix A.4). This forest mask establishes a higher resolution than previous forest masks (10-meter resolution) and is based on imagery of 2020, making it more up-to-date than previous forest masks. Data from all over the world can be used as input to the model. However, as the climate conditions of the input data become more different from those in Germany, the performance of the model most likely decreases. Monitoring the ecosystem becomes easier with the use of this forest mask. On a higher note, by managing the ecosystem accordingly, consequences of climate change can be mitigated.

To further enhance the robustness of this study, several key contributions have been made:

- A base model has been provided that effectively segments forests using aggregated satellite data.
- An algorithm of sequence selection suitable for remote sensed data has been conceptualized and implemented, addressing issues introduced by cloud cover and faulty satellite data.
- A novel modification to the U-Net architecture, the M2O U-Net, has been introduced. This architecture establishes a comprehensive understanding of the ground truth from sequential data using a Fusion Block, with experiments showing results similar to the base model with aggregated data.
- ELR has been applied to the field of semantic segmentation, with various experiments conducted to evaluate its effects. These experiments show ELR to mitigate memorization effectively, but leave question to its effect with early stopping.

Overall, this study not only provides a valuable tool for monitoring the ecosystem in Germany but also provides resources for further research in the interdisciplinary field bridging machine learning and remote sensing.

A Appendix

A.1 Early Stopping in detail

Algorithm 2 Early Stopping [GBC16, p.244]

Input: Initial parameters θ_0, number of steps between evaluations n, patience p
Initialize:
$\theta \leftarrow \theta_0$, $i \leftarrow 0$, $j \leftarrow 0$, $v \leftarrow \infty$ ▷ variables for operation
$\theta^* \leftarrow \theta$, $i^* \leftarrow i$ ▷ variables to save optimum
while $j < p$ **do**
    Update θ by running the training algorithm for n steps.
    $i \leftarrow i + n$
    $v' \leftarrow \mathcal{L}_{X^{(val)}}(\theta)$
    if $v' < v$ **then**
        $j \leftarrow 0$
        $\theta^* \leftarrow \theta$
        $i^* \leftarrow i$
        $v \leftarrow v'$
    else
        $j \leftarrow j + 1$
return θ^*, i^*

Algorithm 2 illustrates the early stopping procedure. It starts with initial weights θ_0, training steps between evaluations n, and a patience parameter p. Key variables are initialized: θ for current parameters, i for total training steps, j for the patience counter, and v for the best validation error, initially set to infinity. The best weight set θ^* and steps i^* are also stored. The algorithm updates θ by training for n steps, increments i, and calculates the validation error v'. If v' is better than v, j resets to 0, and θ^*, i^*, and v are updated. If v' does not improve, j increments. This continues until j reaches the patience p, indicating no improvement for p evaluations. The best parameters θ^* and training steps i^* are then returned.

A.2 Alternative Attempt: Review

The incorporation of the Fusion Block at the skip-connections of the U-Net shows worse performance than the base model (Section 5). Figure 25 clearly shows how using the aggregated sample on the original U-Net leads to less loss right from the start. With longer sequences, the model results in higher losses, providing the assumption that increased use of the Fusion Block limits performance[9]. This assumption is reinforced by the results seen in Figure 26. More revisits used for the segmentation task

[9] Remember that with longer sequences, more iterations through the Fusion Block are necessary to fuse it into a scalar shape, squeezing the dimension of revisits.

generally result in worse pixel accuracy. Only the $R = 2$ case closely resembles the accuracy of the base model; this, however, cannot be credited to the higher information quality provided by multiple timestamps.

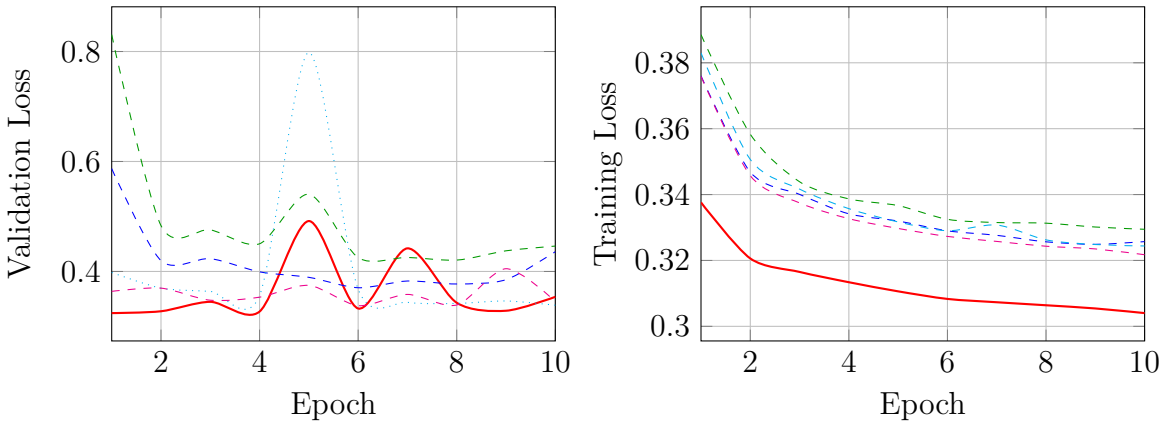


Figure 25 Loss comparison for different sequence lengths

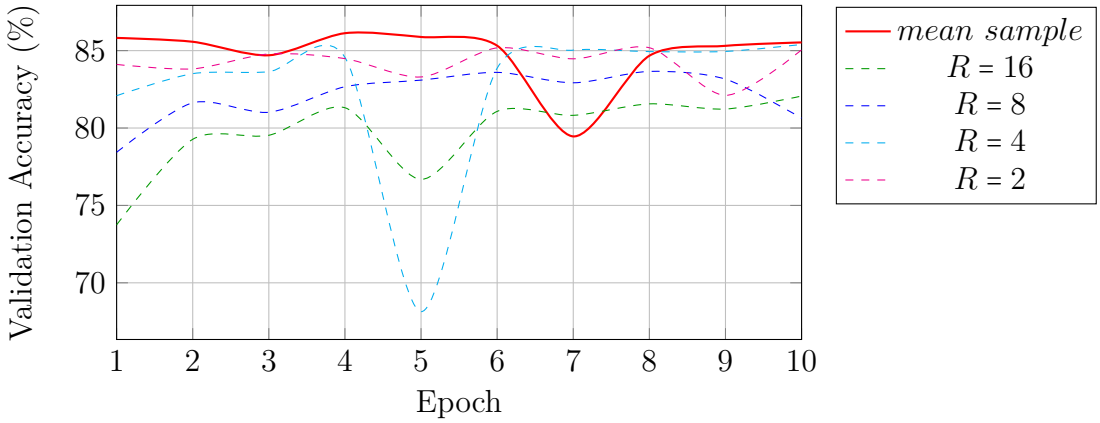


Figure 26 Validation Accuracy (%) for different sequence lengths

A.3 Early Learning Regularization for Semantic Segmentation with Binary Labels

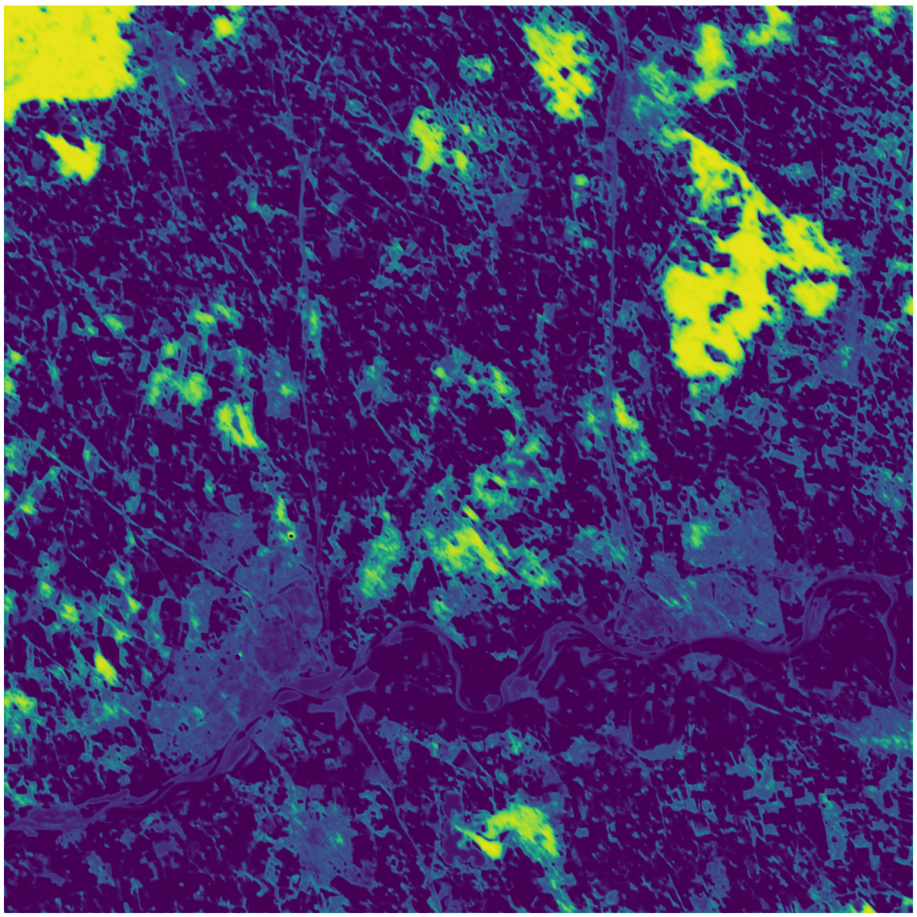
Code 1 Code extract showing the ELR forward pass for semantic segmentation with binary labels.

```
def forward(self, index, output, label):
# Check if CUDA (GPU support) is available and move tensors to GPU if so
if self.USE_CUDA:
output = output.cuda()  # Move output tensor to GPU
label = label.cuda()    # Move label tensor to GPU
index = torch.tensor(index).cuda()  # Convert index to tensor and move to GPU
# Clamp the output values to avoid extreme values
y_pred = torch.clamp(output, 1e-4, 1.0 - 1e-4)
y_pred_ = y_pred.data.detach()  # Detach the tensor from the computational graph
# Update the target using a running average with the current prediction
self.target[index] = self.beta * self.target[index] + (1 - self.beta) * y_pred_.mean()
# Calculate the binary cross-entropy loss between prediction and label
ce_loss = F.binary_cross_entropy(y_pred, label)
# Calculate the ELR (Early Learning Regularization) term
elr_reg = (
(1 - (self.target[index] * y_pred +
(1 - self.target[index]) * (1 - y_pred)))
.log()
).mean()
# Combine the cross-entropy loss and the ELR term to get the final loss
final_loss = ce_loss + self.lamb * elr_reg
return final_loss
```

The *forward* method implements Early Learning Regularization for binary semantic segmentation. It first checks for CUDA availability, moving tensors to the GPU if available. The output tensor is clamped to avoid extreme values and then detached from the computational graph. The target is updated using an exponential moving average with parameter β. Binary cross-entropy loss between the predictions and true labels is calculated, followed by the ELR term, which regularizes early learning. The final loss, a combination of binary cross-entropy and ELR, is returned for backpropagation 1.

A.4 Large Scale Forest Mask

Figure 27 shows a large image, which was propagated through the base model as a single input. The model output shows accurate forest segmentation unaffected by the input scale. Therefore, with no computational limitations, detailed forest masks can be achieved on any scale.



(b) Model output on scale [0,1]

(a) Sentinel-2 rgb converted image data

Figure 27 Large scale forest mask based on Sentinel-2 data

Bibliography

[AD05] S. Aggarwal and Dehra Dun. "PRINCIPLES OF REMOTE SENSING". In: 2005. (Visited on 06/11/2024).

[Aga19] Abien Fred Agarap. *Deep Learning Using Rectified Linear Units (ReLU)*. Feb. 2019. DOI: 10.48550/arXiv.1803.08375. arXiv: 1803.08375 [cs, stat]. (Visited on 07/31/2024).

[AL21] Hussain Alibrahim and Simone A. Ludwig. "Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization". In: *2021 IEEE Congress on Evolutionary Computation (CEC)*. Kraków, Poland: IEEE, June 2021, pp. 1551–1559. ISBN: 978-1-72818-393-0. DOI: 10.1109/CEC45853.2021.9504761. (Visited on 06/07/2024).

[AR19] Assaf Arbelle and Tammy Riklin Raviv. *Microscopy Cell Segmentation via Convolutional LSTM Networks*. Jan. 2019. DOI: 10.48550/arXiv.1805.11247. arXiv: 1805.11247 [cs]. (Visited on 06/05/2024).

[Arp+17] Devansh Arpit et al. "A Closer Look at Memorization in Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, July 2017, pp. 233–242. (Visited on 05/22/2024).

[Ban+14] Asim Banskota et al. "Forest Monitoring Using Landsat Time Series Data: A Review". In: *Canadian Journal of Remote Sensing* 40.5 (Sept. 2014), pp. 362–384. ISSN: 0703-8992, 1712-7971. DOI: 10.1080/07038992.2014.987376. (Visited on 06/25/2024).

[BB12] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13.10 (2012), pp. 281–305. ISSN: 1533-7928. (Visited on 06/22/2024).

[Ben12] Yoshua Bengio. "Deep Learning of Representations for Unsupervised and Transfer Learning". In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. JMLR Workshop and Conference Proceedings, June 2012, pp. 17–36. (Visited on 06/18/2024).

[Bis95] Christopher M Bishop. "Neural Networks for Pattern Recognition". In: (1995).

[BSG15] Matthias Bürgi, Daniel Salzmann, and Urs Gimmi. "264 Years of Change and Persistence in an Agrarian Landscape: A Case Study from the Swiss Lowlands". In: *Landscape Ecology* 30.7 (Aug. 2015), pp. 1321–1333. ISSN: 1572-9761. DOI: 10.1007/s10980-015-0189-1. (Visited on 06/24/2024).

[CCF19] Daniel Cunningham, Paul Cunningham, and Matthew E. Fagan. "Identifying Biases in Global Tree Cover Products: A Case Study in Costa Rica". In: *Forests* 10.10 (Oct. 2019), p. 853. ISSN: 1999-4907. DOI: 10.3390/f10100853. (Visited on 07/23/2024).

[CD15] Marc Claesen and Bart De Moor. *Hyperparameter Search in Machine Learning*. Apr. 2015. DOI: 10.48550/arXiv.1502.02127. arXiv: 1502.02127 [cs, stat]. (Visited on 05/29/2024).

[Cha+14] Jérôme Chave et al. "Improved Allometric Models to Estimate the Aboveground Biomass of Tropical Trees". In: *Global Change Biology* 20.10 (2014), pp. 3177–3190. ISSN: 1365-2486. DOI: 10.1111/gcb.12629. (Visited on 02/28/2024).

[Che+17] Liang-Chieh Chen et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. May 2017. DOI: 10.48550/arXiv.1606.00915. arXiv: 1606.00915 [cs]. (Visited on 06/07/2024).

[Cho+20] Dami Choi et al. *On Empirical Comparisons of Optimizers for Deep Learning*. June 2020. arXiv: 1910.05446 [cs, stat]. (Visited on 06/08/2024).

[Cho24] Yeong-Jun Cho. *Weighted Intersection over Union (wIoU) for Evaluating Image Segmentation*. July 2024. arXiv: 2107.09858 [cs]. (Visited on 07/04/2024).

[Chr+19] Vincent Christlein et al. "Deep Generalized Max Pooling". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pp. 1090–1096. DOI: 10.1109/ICDAR.2019.00177. arXiv: 1908.05040 [cs]. (Visited on 07/31/2024).

[CJ22] Davide Chicco and Giuseppe Jurman. "An Invitation to Greater Use of Matthews Correlation Coefficient in Robotics and Artificial Intelligence". In: *Frontiers in Robotics and AI* 9 (Mar. 2022). ISSN: 2296-9144. DOI: 10.3389/frobt.2022.876814. (Visited on 07/31/2024).

[COK22] Julien Cornebise, Ivan Oršolić, and Freddie Kalaitzis. "Open High-Resolution Satellite Imagery: The WorldStrat Dataset – With Application to Super-Resolution". In: *Advances in Neural Information Processing Systems* 35 (Dec. 2022), pp. 25979–25991. (Visited on 06/05/2024).

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. (Visited on 06/11/2024).

[CW11] James B. Campbell and Randolph H. Wynne. *Introduction to Remote Sensing, Fifth Edition*. Guilford Press, June 2011. ISBN: 978-1-60918-177-2.

[Dad+02] V K Dadhwal et al. "Remote Sensing Based Crop Inventory: A Review of Indian Experience". In: (2002).

[Den+09] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Miami, FL: IEEE, June 2009, pp. 248–255. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206848. (Visited on 06/06/2024).

[Den12] Li Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]". In: *IEEE Signal Processing Magazine* 29.6 (Nov. 2012), pp. 141–142. ISSN: 1558-0792. DOI: 10.1109/MSP.2012.2211477. (Visited on 07/30/2024).

[Die95] Tom Dietterich. "Overfitting and Undercomputing in Machine Learning". In: *ACM Computing Surveys* 27.3 (Sept. 1995), pp. 326–327. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/212094.212114. (Visited on 06/19/2024).

[Du+20] Xianzhi Du et al. *SpineNet: Learning Scale-Permuted Backbone for Recognition and Localization*. 2020. arXiv: 1912.05027 [cs.CV].

[Fam+97] A Famili et al. "Data Preprocessing and Intelligent Data Analysis". In: *Intelligent Data Analysis* 1.1-4 (1997), pp. 3–23. ISSN: 1088467X. DOI: 10.1016/S1088-467X(98)00007-9. (Visited on 05/28/2024).

[Faw04] Tom Fawcett. "ROC Graphs: Notes and Practical Considerations for Researchers". In: *Machine Learning* 31 (Jan. 2004), pp. 1–38.

[Fer+18] Eduardo Fernandez-Moral et al. "A New Metric for Evaluating Semantic Segmentation: Leveraging Global and Contour Accuracy". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu: IEEE, June 2018, pp. 1051–1056. ISBN: 978-1-5386-4452-2. DOI: 10.1109/IVS.2018.8500497. (Visited on 07/04/2024).

[Fid+18] Lucas Fidon et al. "Generalised Wasserstein Dice Score for Imbalanced Multi-class Segmentation Using Holistic Convolutional Networks". In: vol. 10670. 2018, pp. 64–76. DOI: 10.1007/978-3-319-75238-9_6. arXiv: 1707.00478 [cs]. (Visited on 07/25/2024).

[FV14] Benoît Frénay and Michel Verleysen. "Classification in the Presence of Label Noise: A Survey". In: *Neural Networks and Learning Systems, IEEE Transactions on* 25 (May 2014), pp. 845–869. DOI: 10.1109/TNNLS.2013.2292894.

[Gar+17] Alberto Garcia-Garcia et al. *A Review on Deep Learning Techniques Applied to Semantic Segmentation*. Apr. 2017. DOI: 10.48550/arXiv.1704.06857. arXiv: 1704.06857 [cs]. (Visited on 06/18/2024).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GIS19] GISGeography. *Sentinel 2 Bands and Combinations*. Apr. 2019. URL: https://gisgeography.com/sentinel-2-bands-combinations/ (visited on 05/27/2024).

[GK20] Hossein Gholamalinezhad and Hossein Khosravi. *Pooling Methods in Deep Neural Networks, a Review*. Sept. 2020. DOI: 10.48550/arXiv.2009.07485. arXiv: 2009.07485 [cs]. (Visited on 06/17/2024).

[Goo+14] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014. (Visited on 06/18/2024).

[Han+13] M.C. Hansen et al. "High-Resolution Global Maps of 21st-Century Forest Cover Change". In: *Science (New York, N.Y.)* 342 (Nov. 2013), pp. 850–853. DOI: 10.1126/science.1244693.

[Han+18] Bo Han et al. *Co-Teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels*. Oct. 2018. DOI: 10.48550/arXiv.1804.06872. arXiv: 1804.06872 [cs, stat]. (Visited on 05/17/2024).

[Han+21] Bo Han et al. *A Survey of Label-noise Representation Learning: Past, Present and Future*. Feb. 2021. DOI: 10.48550/arXiv.2011.04406. arXiv: 2011.04406 [cs]. (Visited on 05/04/2024).

[Has19] Mahdi Hashemi. "Enlarging Smaller Images before Inputting into Convolutional Neural Network: Zero-Padding vs. Interpolation". In: *Journal of Big Data* 6.1 (Nov. 2019), p. 98. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0263-7. (Visited on 07/31/2024).

[HB21] Like Hui and Mikhail Belkin. *Evaluation of Neural Architectures Trained with Square Loss vs Cross-Entropy in Classification Tasks*. Oct. 2021. DOI: 10.48550/arXiv.2006.07322. arXiv: 2006.07322 [cs, stat]. (Visited on 07/30/2024).

[He+15a] Kaiming He et al. *Deep Residual Learning for Image Recognition*. Dec. 2015. DOI: 10.48550/arXiv.1512.03385. arXiv: 1512.03385 [cs]. (Visited on 05/28/2024).

[He+15b] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. Feb. 2015. DOI: 10.48550/arXiv.1502.01852. arXiv: 1502.01852 [cs]. (Visited on 07/09/2024).

[He+16] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. July 2016. DOI: 10.48550/arXiv.1603.05027. arXiv: 1603.05027 [cs]. (Visited on 05/27/2024).

[He+18] Kaiming He et al. *Mask R-CNN*. Jan. 2018. DOI: 10.48550/arXiv.1703.06870. arXiv: 1703.06870 [cs]. (Visited on 06/07/2024).

[Hec89] Hecht-Nielsen. "Theory of the Backpropagation Neural Network". In: *International 1989 Joint Conference on Neural Networks*. June 1989, 593–605 vol.1. DOI: 10.1109/IJCNN.1989.118638. (Visited on 06/13/2024).

[HGD18] Kaiming He, Ross Girshick, and Piotr Dollár. *Rethinking ImageNet Pre-training*. Nov. 2018. DOI: 10.48550/arXiv.1811.08883. arXiv: 1811.08883 [cs]. (Visited on 05/29/2024).

[Hic96] Ray J. Hickey. "Noise Modelling and Evaluating Learning from Examples". In: *Artificial Intelligence* 82.1 (Apr. 1996), pp. 157–179. ISSN: 0004-3702. DOI: 10.1016/0004-3702(94)00094-8. (Visited on 05/29/2024).

[HKN19] Chongomweru Halimu, Asem Kasem, and S. H. Shah Newaz. "Empirical Comparison of Area under ROC Curve (AUC) and Mathew Correlation Coefficient (MCC) for Evaluating Machine Learning Algorithms on Imbalanced Datasets for Binary Classification". In: *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing*. ICMLSC '19. New York, NY, USA: Association for Computing Machinery, Jan. 2019, pp. 1–6. ISBN: 978-1-4503-6612-0. DOI: 10.1145/3310986.3311023. (Visited on 07/04/2024).

[HKP11] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Elsevier, June 2011. ISBN: 978-0-12-381480-7.

[Ho95] Tin Kam Ho. "Random Decision Forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. Aug. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994. (Visited on 06/11/2024).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[HT12] Geoffrey Hinton and Tijmen Tieleman. *Lecture 6.5-Rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude*. 2012.

[HT21] Hong Hai Hoang and Hoang Hieu Trinh. "Improvement for Convolutional Neural Networks in Image Classification Using Long Skip Connection". In: *Applied Sciences* 11.5 (Jan. 2021), p. 2092. ISSN: 2076-3417. DOI: 10.3390/app11052092. (Visited on 07/31/2024).

[HW20] Yaoshiang Ho and Samuel Wookey. "The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling". In: *IEEE Access* 8 (2020), pp. 4806–4813. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2962617. (Visited on 07/30/2024).

[IS15] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. Mar. 2015. DOI: 10.48550/arXiv.1502.03167. arXiv: 1502.03167 [cs]. (Visited on 06/19/2024).

[Jad20] Shruti Jadon. "A Survey of Loss Functions for Semantic Segmentation". In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. Oct. 2020, pp. 1–7. DOI: 10.1109/CIBCB48159.2020.9277638. arXiv: 2006.14822 [cs, eess]. (Visited on 07/04/2024).

[Jia+18] Lu Jiang et al. *MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels*. Aug. 2018. DOI: 10.48550/arXiv.1712.05055. arXiv: 1712.05055 [cs]. (Visited on 05/19/2024).

[KB17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980 [cs]. (Visited on 06/08/2024).

[KGC17] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. *Regularization for Deep Learning: A Taxonomy*. Oct. 2017. DOI: 10.48550/arXiv.1710.10686. arXiv: 1710.10686 [cs, stat]. (Visited on 06/18/2024).

[KH91] Anders Krogh and John Hertz. "A Simple Weight Decay Can Improve Generalization". In: *Advances in Neural Information Processing Systems*. Vol. 4. Morgan-Kaufmann, 1991. (Visited on 06/19/2024).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012. (Visited on 06/18/2024).

[LA17] Samuli Laine and Timo Aila. *Temporal Ensembling for Semi-Supervised Learning*. Mar. 2017. arXiv: 1610.02242 [cs]. (Visited on 06/21/2024).

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. (Visited on 06/12/2024).

[LeC+12] Yann A. LeCun et al. “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer, 2012, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3. (Visited on 06/21/2024).

[LeC+89] Y. LeCun et al. “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* 1.4 (Dec. 1989), pp. 541–551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541. (Visited on 06/16/2024).

[LeC+98] Yann LeCun et al. “Gradient-Based Learning Applied to Document Recognition”. In: (1998).

[LH91] H. Leung and S. Haykin. “The Complex Backpropagation Algorithm”. In: *IEEE Transactions on Signal Processing* 39.9 (Sept. 1991), pp. 2101–2104. ISSN: 1941-0476. DOI: 10.1109/78.134446. (Visited on 06/13/2024).

[Li+18] Ying Li et al. “Deep Learning for Remote Sensing Image Classification: A Survey”. In: *WIREs Data Mining and Knowledge Discovery* 8.6 (2018), e1264. ISSN: 1942-4795. DOI: 10.1002/widm.1264. (Visited on 06/11/2024).

[Li+20a] Hao Li et al. *Rethinking the Hyperparameters for Fine-tuning*. Feb. 2020. DOI: 10.48550/arXiv.2002.11770. arXiv: 2002.11770 [cs, stat]. (Visited on 05/29/2024).

[Li+20b] Zewen Li et al. *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects*. Apr. 2020. DOI: 10.48550/arXiv.2004.02806. arXiv: 2004.02806 [cs, eess]. (Visited on 06/16/2024).

[Liu+19] Zhenqing Liu et al. “Computer Vision-Based Concrete Crack Detection Using U-net Fully Convolutional Networks”. In: *Automation in Construction* 104 (Aug. 2019), pp. 129–139. ISSN: 0926-5805. DOI: 10.1016/j.autcon.2019.04.005. (Visited on 06/18/2024).

[Liu+20] Sheng Liu et al. “Early-Learning Regularization Prevents Memorization of Noisy Labels”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 20331–20342. (Visited on 02/28/2024).

[LLZ19] Siyuan Lu, Zhihai Lu, and Yu-Dong Zhang. “Pathological Brain Detection Based on AlexNet and Transfer Learning”. In: *Journal of Computational Science* 30 (Jan. 2019), pp. 41–47. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2018.11.008. (Visited on 06/18/2024).

[Lu+20] Lu Lu et al. "Dying ReLU and Initialization: Theory and Numerical Examples". In: *Communications in Computational Physics* 28.5 (June 2020), pp. 1671–1706. ISSN: 1815-2406, 1991-7120. DOI: 10.4208/cicp.OA-2020-0165. arXiv: 1903.06733 [cs, math, stat]. (Visited on 07/31/2024).

[Ma+15] Yan Ma et al. "Remote Sensing Big Data Computing: Challenges and Opportunities". In: *Future Generation Computer Systems* 51 (Oct. 2015), pp. 47–60. ISSN: 0167739X. DOI: 10.1016/j.future.2014.10.029. (Visited on 06/11/2024).

[Ma+19] Lei Ma et al. "Deep Learning in Remote Sensing Applications: A Meta-Analysis and Review". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 152 (June 2019), pp. 166–177. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2019.04.015. (Visited on 05/24/2024).

[MCJ06] David McClosky, Eugene Charniak, and Mark Johnson. "Effective Self-Training for Parsing". In: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. Ed. by Robert C. Moore et al. New York City, USA: Association for Computational Linguistics, June 2006, pp. 152–159. (Visited on 06/21/2024).

[Min+20] Shervin Minaee et al. *Image Segmentation Using Deep Learning: A Survey*. Nov. 2020. DOI: 10.48550/arXiv.2001.05566. arXiv: 2001.05566 [cs]. (Visited on 06/18/2024).

[MNA16] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*. June 2016. DOI: 10.48550/arXiv.1606.04797. arXiv: 1606.04797 [cs]. (Visited on 06/19/2024).

[Mun+14] Catalina Munteanu et al. "Forest and Agricultural Land Change in the Carpathian Region—A Meta-Analysis of Long-Term Patterns and Drivers of Change". In: *Land Use Policy* 38 (May 2014), pp. 685–697. DOI: 10.1016/j.landusepol.2014.01.012.

[Ngi+18] Jiquan Ngiam et al. *Domain Adaptive Transfer Learning with Specialist Models*. Dec. 2018. DOI: 10.48550/arXiv.1811.07056. arXiv: 1811.07056 [cs]. (Visited on 06/07/2024).

[NVR07] Ranganath Navalgund, Jayaraman V, and Parth Roy. "Remote Sensing Applications: An Overview". In: *Current science* Vol. 93 (Dec. 2007).

[ON15] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. Dec. 2015. DOI: 10.48550/arXiv.1511.08458. arXiv: 1511.08458 [cs]. (Visited on 06/16/2024).

[Ong17] Pariwat Ongsulee. “Artificial Intelligence, Machine Learning and Deep Learning”. In: *2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE)*. Nov. 2017, pp. 1–6. DOI: 10.1109/ICTKE.2017.8259629. (Visited on 07/30/2024).

[Osc+21] Lucas Prado Osco et al. “A Review on Deep Learning in UAV Remote Sensing”. In: *International Journal of Applied Earth Observation and Geoinformation* 102 (Oct. 2021), p. 102456. ISSN: 1569-8432. DOI: 10.1016/j.jag.2021.102456. (Visited on 06/11/2024).

[Pan+11] Yude Pan et al. “A Large and Persistent Carbon Sink in the World’s Forests”. In: *Science (New York, N.Y.)* 333 (Aug. 2011), pp. 988–93. DOI: 10.1126/science.1201609.

[Pau+24] Jan Pauls et al. *Estimating Canopy Height at Scale*. June 2024. arXiv: 2406.01076 [cs]. (Visited on 06/24/2024).

[PS15] S.Gopal Krishna Patro and Kishore Kumar Sahu. “Normalization: A Preprocessing Stage”. In: *IARJSET* (Mar. 2015), pp. 20–22. ISSN: 23938021. DOI: 10.17148/IARJSET.2015.2305. (Visited on 07/31/2024).

[PTH11] Peter Potapov, Svetlana Turubanova, and Matthew C. Hansen. “Regional-Scale Boreal Forest Cover and Change Mapping Using Landsat Data Composites for European Russia”. In: *Remote Sensing of Environment* 115.2 (Feb. 2011), pp. 548–561. ISSN: 0034-4257. DOI: 10.1016/j.rse.2010.10.001. (Visited on 06/26/2024).

[Qi+20] Wenwen Qi et al. “Automatic Mapping of Landslides by the ResU-Net”. In: *Remote Sensing* 12.15 (Jan. 2020), p. 2487. ISSN: 2072-4292. DOI: 10.3390/rs12152487. (Visited on 06/06/2024).

[Qia99] Ning Qian. “On the Momentum Term in Gradient Descent Learning Algorithms”. In: *Neural Networks* 12.1 (Jan. 1999), pp. 145–151. ISSN: 08936080. DOI: 10.1016/S0893-6080(98)00116-6. (Visited on 06/09/2024).

[Ren+16] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Jan. 2016. DOI: 10.48550/arXiv.1506.01497. arXiv: 1506.01497 [cs]. (Visited on 06/07/2024).

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4. DOI: 10.1007/978-3-319-24574-4_28.

[Rud17] Sebastian Ruder. *An Overview of Gradient Descent Optimization Algorithms*. June 2017. DOI: 10.48550/arXiv.1609.04747. arXiv: 1609.04747 [cs]. (Visited on 06/08/2024).

[Rum+95] D. Rumelhart et al. "Backpropagation: The Basic Theory". In: 1995. (Visited on 06/14/2024).

[San+20] Alber Hamersson Sanchez et al. "Comparison of Cloud Cover Detection Algorithms on Sentinel–2 Images of the Amazon Tropical Forest". In: *Remote Sensing* 12.8 (Jan. 2020), p. 1284. ISSN: 2072-4292. DOI: 10.3390/rs12081284. (Visited on 06/26/2024).

[SBP05] Anatoly Shvidenko, C.V. Barber, and R. Persson. "Forests and Woodland Systems". In: *Ecosystems and Human Well-being: Current State and Trends* 1 (Jan. 2005), pp. 587–621.

[Sch+21] Lisa Schneider et al. "Exploring Bias in F-score Computation Methods of Multi-Class Segmentation Models". In: *2021 The 5th International Conference on Video and Image Processing*. Hayward CA USA: ACM, Dec. 2021, pp. 76–84. ISBN: 978-1-4503-8589-3. DOI: 10.1145/3511176.3511189. (Visited on 07/04/2024).

[Seo+20] Hyunseok Seo et al. "Modified U-Net (mU-Net) With Incorporation of Object-Dependent High Level Features for Improved Liver and Liver-Tumor Segmentation in CT Images". In: *IEEE Transactions on Medical Imaging* 39.5 (May 2020), pp. 1316–1325. ISSN: 1558-254X. DOI: 10.1109/TMI.2019.2948320. (Visited on 06/18/2024).

[Sha+19] Pourya Shamsolmoali et al. "A Novel Deep Structure U-Net for Sea-Land Segmentation in Remote Sensing Images". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.9 (Sept. 2019), pp. 3219–3232. ISSN: 1939-1404, 2151-1535. DOI: 10.1109/JSTARS.2019.2925841. (Visited on 06/06/2024).

[SIN89] ASHBINDU SINGH. "Review Article Digital Change Detection Techniques Using Remotely-Sensed Data". In: *International Journal of Remote Sensing* 10.6 (June 1989), pp. 989–1003. ISSN: 0143-1161. DOI: 10.1080/01431168908903939. (Visited on 06/25/2024).

[Son+22] Hwanjun Song et al. *Learning from Noisy Labels with Deep Neural Networks: A Survey*. Mar. 2022. DOI: 10.48550/arXiv.2007.08199. arXiv: 2007.08199 [cs, stat]. (Visited on 05/04/2024).

[Sri+14] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. ISSN: 1533-7928. (Visited on 06/19/2024).

[SS19] T. Shanthi and R. S. Sabeenian. “Modified Alexnet Architecture for Classification of Diabetic Retinopathy Images”. In: *Computers & Electrical Engineering* 76 (June 2019), pp. 56–64. ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2019.03.004. (Visited on 06/18/2024).

[SY16] Patrick T. Sekoai and Kelvin O. Yoro. “Biofuel Development Initiatives in Sub-Saharan Africa: Opportunities and Challenges”. In: *Climate* 4.2 (June 2016), p. 33. ISSN: 2225-1154. DOI: 10.3390/cli4020033. (Visited on 06/24/2024).

[Tia+22] Yingjie Tian et al. “Recent Advances on Loss Functions in Deep Learning for Computer Vision”. In: *Neurocomputing* 497 (Aug. 2022), pp. 129–158. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.04.127. (Visited on 06/12/2024).

[Tib96] Robert Tibshirani. “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 0035-9246. JSTOR: 2346178. (Visited on 06/21/2024).

[Tro+14] Robert Tropek et al. “Comment on “High-resolution Global Maps of 21st-Century Forest Cover Change””. In: *Science* 344.6187 (May 2014), pp. 981–981. DOI: 10.1126/science.1248753. (Visited on 07/23/2024).

[Van+21] Elia Vangi et al. “The Effect of Forest Mask Quality in the Wall-to-Wall Estimation of Growing Stock Volume”. In: *Remote Sensing* 13.5 (Jan. 2021), p. 1038. ISSN: 2072-4292. DOI: 10.3390/rs13051038. (Visited on 07/30/2024).

[WBK23] Piper Wolters, Favyen Bastani, and Aniruddha Kembhavi. *Zooming Out on Zooming In: Advancing Super-Resolution for Remote Sensing*. Nov. 2023. arXiv: 2311.18082 [cs]. (Visited on 02/28/2024).

[Wei66] Joseph Weizenbaum. “ELIZA—a Computer Program for the Study of Natural Language Communication between Man and Machine”. In: *Communications of the ACM* 9.1 (Jan. 1966), pp. 36–45. ISSN: 0001-0782. DOI: 10.1145/365153.365168. (Visited on 06/11/2024).

[Wer90] P.J. Werbos. “Backpropagation through Time: What It Does and How to Do It”. In: *Proceedings of the IEEE* 78.10 (Oct. 1990), pp. 1550–1560. ISSN: 1558-2256. DOI: 10.1109/5.58337. (Visited on 06/13/2024).

[WG15] Haibing Wu and Xiaodong Gu. “Towards Dropout Training for Convolutional Neural Networks”. In: *Neural Networks* 71 (Nov. 2015), pp. 1–10. ISSN: 08936080. DOI: 10.1016/j.neunet.2015.07.007. arXiv: 1512.00242 [cs]. (Visited on 06/17/2024).

[Wil+18] Ashia C. Wilson et al. *The Marginal Value of Adaptive Gradient Methods in Machine Learning*. May 2018. DOI: 10.48550/arXiv.1705.08292. arXiv: 1705.08292 [cs, stat]. (Visited on 06/08/2024).

[WKW16] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. "A Survey of Transfer Learning". In: *Journal of Big Data* 3.1 (May 2016), p. 9. ISSN: 2196-1115. DOI: 10.1186/s40537-016-0043-6. (Visited on 06/18/2024).

[WM22] Haiying Wang and Fang Miao. "Building Extraction from Remote Sensing Images Using Deep Residual U-Net". In: *European Journal of Remote Sensing* 55.1 (Dec. 2022), pp. 71–85. ISSN: null. DOI: 10.1080/22797254.2021.2018944. (Visited on 06/06/2024).

[WTJ21] Ross Wightman, Hugo Touvron, and Hervé Jégou. *ResNet Strikes Back: An Improved Training Procedure in Timm*. Oct. 2021. arXiv: 2110.00476 [cs]. (Visited on 08/01/2024).

[Yan+14] Yuanyuan Yang et al. "A Review of Historical Reconstruction Methods of Land Use/Land Cover". In: *Journal of Geographical Sciences* 24.4 (Aug. 2014), pp. 746–766. ISSN: 1861-9568. DOI: 10.1007/s11442-014-1117-z. (Visited on 06/24/2024).

[YD20] Kelvin O. Yoro and Michael O. Daramola. "CO2 Emission Sources, Greenhouse Gases, and the Global Warming Effect". In: *Advances in Carbon Capture*. Ed. by Mohammad Reza Rahimpour, Mohammad Farsi, and Mohammad Amin Makarem. Woodhead Publishing, Jan. 2020, pp. 3–28. ISBN: 978-0-12-819657-1. DOI: 10.1016/B978-0-12-819657-1.00001-3. (Visited on 06/24/2024).

[Yos+14] Jason Yosinski et al. *How Transferable Are Features in Deep Neural Networks?* Nov. 2014. DOI: 10.48550/arXiv.1411.1792. arXiv: 1411.1792 [cs]. (Visited on 06/18/2024).

[YS20] Li Yang and Abdallah Shami. "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice". In: *Neurocomputing* 415 (Nov. 2020), pp. 295–316. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2020.07.061. (Visited on 06/08/2024).

[Zaf+22] Afia Zafar et al. "A Comparison of Pooling Methods for Convolutional Neural Networks". In: *Applied Sciences* 12.17 (Jan. 2022), p. 8643. ISSN: 2076-3417. DOI: 10.3390/app12178643. (Visited on 06/17/2024).

[Zha+17] Chiyuan Zhang et al. *Understanding Deep Learning Requires Rethinking Generalization*. Feb. 2017. DOI: 10.48550/arXiv.1611.03530. arXiv: 1611.03530 [cs]. (Visited on 05/19/2024).

[Zho+17] Lina Zhou et al. “Machine Learning on Big Data: Opportunities and Challenges”. In: *Neurocomputing* 237 (May 2017), pp. 350–361. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.01.026. (Visited on 07/30/2024).

[Zhu+17] Xiao Xiang Zhu et al. “Deep Learning in Remote Sensing: A Review”. In: *IEEE Geoscience and Remote Sensing Magazine* 5.4 (Dec. 2017), pp. 8–36. ISSN: 2168-6831, 2473-2397, 2373-7468. DOI: 10.1109/MGRS.2017.2762307. arXiv: 1710.03959 [cs, eess]. (Visited on 05/23/2024).

[ZLW18] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. “Road Extraction by Deep Residual U-Net”. In: *IEEE Geoscience and Remote Sensing Letters* 15.5 (May 2018), pp. 749–753. ISSN: 1545-598X, 1558-0571. DOI: 10.1109/LGRS.2018.2802944. (Visited on 06/06/2024).

[Zop+20] Barret Zoph et al. *Rethinking Pre-training and Self-training*. Nov. 2020. DOI: 10.48550/arXiv.2006.06882. arXiv: 2006.06882 [cs, stat]. (Visited on 05/29/2024).

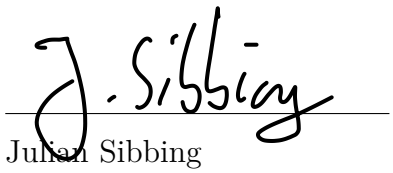
[ZW04] Xingquan Zhu and Xindong Wu. “Class Noise vs. Attribute Noise: A Quantitative Study”. In: *Artificial Intelligence Review* 22.3 (Nov. 2004), pp. 177–210. ISSN: 1573-7462. DOI: 10.1007/s10462-004-0751-8. (Visited on 05/29/2024).

[ZZD16] Liangpei Zhang, Lefei Zhang, and Bo Du. “Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art”. In: *IEEE Geoscience and Remote Sensing Magazine* 4.2 (June 2016), pp. 22–40. ISSN: 2168-6831. DOI: 10.1109/MGRS.2016.2540798. (Visited on 03/05/2024).

Declaration of Authorship

I hereby declare that, to the best of my knowledge and belief, this thesis titled *Advancements in Remote Sensing: Developing an Updated Forest Mask through Deep Learning* is my own, independent work. I confirm that each significant contribution to and quotation in this thesis that originates from the work or works of others is indicated by proper use of citation and references; this also holds for tables and graphical works.

Münster, 04.08.2024



Julian Sibbing

Consent Form

Name: Julian Sibbing
Title of Thesis: Advancements in Remote Sensing: Developing an Updated Forest Mask through Deep Learning

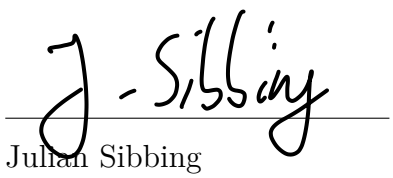
What is plagiarism? Plagiarism is defined as submitting someone else's work or ideas as your own without a complete indication of the source. It is hereby irrelevant whether the work of others is copied word by word without acknowledgment of the source, text structures (e.g. line of argumentation or outline) are borrowed or texts are translated from a foreign language.

Use of plagiarism detection software. The examination office uses plagiarism software to check each submitted bachelor and master thesis for plagiarism. For that purpose the thesis is electronically forwarded to a software service provider where the software checks for potential matches between the submitted work and work from other sources. For future comparisons with other theses, your thesis will be permanently stored in a database. Only the School of Business and Economics of the University of Münster is allowed to access your stored thesis. The student agrees that his or her thesis may be stored and reproduced only for the purpose of plagiarism assessment. The first examiner of the thesis will be advised on the outcome of the plagiarism assessment.

Sanctions Each case of plagiarism constitutes an attempt to deceive in terms of the examination regulations and will lead to the thesis being graded as "failed". This will be communicated to the examination office where your case will be documented. In the event of a serious case of deception the examinee can be generally excluded from any further examination. This can lead to the exmatriculation of the student. Even after completion of the examination procedure and graduation from university, plagiarism can result in a withdrawal of the awarded academic degree.

I confirm that I have read and understood the information in this document. I agree to the outlined procedure for plagiarism assessment and potential sanctioning.

Münster, 04.08.2024



Julian Sibbing