A Case Study on Automated Machine Learning on Binary Image Segmentation for Wildfire Detection for Earth Observation Data

Laura Faas (s3443159)

Supervisors:
J. N. Van Rijn & L. Papa (ESA Φ-lab)

Bachelor Thesis– Informatica

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

August 26, 2026

Abstract

This thesis examines the extent to which automated machine learning improves segmentation of burned area in Earth Observation data. With wildfires increasing both in frequency and intensity, accurate and efficient detection models contribute to fast and reliable alarming systems. Baseline models, such as Random Forest and XGBoost are compared to both an untuned and a tuned U-Net model, to examine their performance. By comparing mIoU scores, insight into the increase of performance through hyperparameter optimization is obtained. To assess the trends in mIoU score, two datasets are compared. Both datasets stem from ESA's Sentinel-2 mission, and the models are trained on this data after binarization.

Results indicate an absolute increase of up to 0.47 in mIoU score of the optimized model compared to the Random Forest model, which was the best performing model of the baselines and 0.02 in mIoU compared to the untuned model. The increase in performance is backed by the second dataset, in which the optimized model scores highest as well. This suggests that automation of hyperparameter optimization can increase performance, while reducing manual optimization, which is both time and labour intensive and requires more expertise. This would make it possible to have well-performing models in regions with limited machine learning resources.

Contents

1 Introduction

Over the last two decades, wildfires have been doubling in size, burning twice as much tree cover as in 2000. The most extreme year this far was 2024, in which roughly 13.5 million hectares were burned [18]. Wildfires pose several risks, affecting both individuals and society. Besides the health problems caused by fire and smoke inhalation [20], wildfires also affect the regional economy and infrastructure significantly. Furthermore, the climate is affected through the release of substantial amounts of greenhouse gases into the atmosphere [11, 9]. With rising temperatures and increased urbanization near rural areas, the risks posed by wildfires are increasing [21]. To reduce damages, communities require strong mitigation strategies, including precise, real-time data to monitor fire progression and alert populations at risk [26]. Therefore, adequate detection systems are crucial. The mapping of burned areas is essential for models to predict wildfires, as they provide a benchmark for comparing predicted wildfires and for measuring accuracy and improving the model. This makes these maps the cornerstone of mitigation strategies [19].

However, the reliability of current detection methods and mappings heavily depends on the methods of collecting data. Historically, ground estimates provided information about burnt areas [9]. This had its limitations; methods varied per country, and data collection was often sporadic, leading to unreliable analysis on a bigger scale. When satellites came into play, some of these limitations were overcome [9]. The now abundant amounts of Earth Observation (EO) data can be used to analyse and predict wildfires through machine learning. Models can extract patterns from this data in order to predict future burned areas — and therefore future wildfires. This requires complex machine learning models like U-Net [30]. However, developing such models is an expensive and labour-intensive undertaking [1]. Engineers need to manually optimize model parameters, such as learning rates, batch sizes, and architectures through trial-and-error processes. Because training these models requires substantial computational resources, manually testing every new configuration quickly becomes a severe bottleneck.

To overcome the labour-intensive process, automated machine learning (AutoML) emerges as a possible solution. AutoML aims to significantly reduce development costs of machine learning models by automating complex and time-consuming design steps, for example through automated hyperparameter optimization (HPO) [1] and neural architecture search (NAS) [33]. It reduces the need for manual intervention, enabling a more scalable, efficient and accessible approach to wildfire monitoring. Scalability is becoming increasingly important in the Earth Observation domain. European Space Agency's most recent artificial intelligence roadmap emphasizes the need for machine learning and artificial intelligence approaches for the growing amount of satellite data for real-time monitoring [30]. AutoML can offer a solution to these challenges. It allows for fast deployment of detection models without the need for extensive manual tuning, providing an efficient method for processing large amounts of satellite data.

However, it remains unclear to what extent AutoML can match or outperform manually tuned models for burned area detection specifically. Therefore, this research presents a case study that investigates to what extent automated machine learning can optimize binary image segmentation for the detection of areas burned by wildfires compared to established baselines, using Earth Observation data. To answer this, baseline models (Majority Class, Random Forest and XGBoost) and a deep-learning model (U-Net) are compared to a U-Net model with tuned hyperparameters through KerasTuner [23]. For this comparison, two semantic segmentation datasets are used, the SegTHRawS [29] and the OEOBench Burnt Area dataset [10]. The results suggest that a tuned

U-Net achieves higher performance in mIoU score.

The thesis is structured as follows. Chapter 2 provides a literature review of existing work on burned area detection and AutoML. Chapter 3 describes the datasets that are used. The models are presented in the theoretical framework of Chapter 4. Chapter 5 elaborates on the methodology, including the experimental design and evaluation metrics. The results are presented in Chapter 6, after which Chapter 7 discusses the findings. Chapter 8 presents the conclusions of this research.

2 Literature Review

The data and methods used in burned area detection have evolved considerably in the last decades [9]. This chapter reviews this evolution from existing approaches to burned area mapping using satellite data, followed by the role of deep learning architectures and the AutoML approaches relevant for this task.

Before satellites came along, burned area detection was based on estimates from fire management teams. Varying methods of collecting these estimates made them unreliable. The arrival of EO imaging made the data more abundant, scalable and reliable [9], though this data still required preprocessing using models that were adapted to different ecosystems manually. Studies in Australia and the US have shown that semi-automated processes can effectively map broad landscapes [13] [5].These studies led to the use of a gradient boosted classifier, which in combination with thresholding led to the BAECV (Burned Area Essential Climate Variable) [15].

The methods used for the BAECV, for example, classified pixels individually without taking spatial context or shapes of burned regions into account. Semantic segmentation methods such as U-Net can factor spatial context and shapes in, learning from the structure directly. In burned area detection it is especially useful, where the question is not only *whether* area is burned, but more importantly *where and in what shape* the area is burned. U-Net, introduced for biomedical image segmentation [25], preserves local details (borders of the shape) and global context (location). For this reason, U-Net has become a standard for segmentation tasks in remote sensing.

A recent application of this approach to thermal hotspot detection is presented by Traba et al. [29]. They use an architecture derived from U-Net, ResUnet-S2, to segment thermal hotspots directly onboard the satellite, using raw Sentinel-2 data. This eliminates the need to send data to Earth for processing. The dataset, SegTHRawS (Segmentation Thermal Hotspots in Raw Sentinel-2 data), is the first publicly available dataset for thermal hotspot segmentation in raw multispectral satellite imagery, according to Traba et al. [29]. It covers 100 events of thermal anomalies, i.e. volcanic eruptions and wildfires. They used one fixed architecture, the ResUnet-S2, to segment thermal hotspots onboard the satellite. However, designing such an architecture by hand is a labour-intensive process. This raises the question of whether an automated process, such as AutoML, can achieve comparable performance without manual optimization efforts.

Although its foundations trace back decades, AutoML is a fairly new research area that tries to make machine learning more accessible by automating the engineering process [1]. Baratchi et al. [1] distinguish four phases in this process: exploring and interpreting the data, preparing the data for a machine learning pipeline, building the model itself through algorithm and hyperparameter selection, and finally applying the resulting model to decision-making. This thesis focuses on hyperparameter optimization (HPO), with a fixed model architecture (U-Net). Such an approach has previously been explored for sea ice concentration charting and achieved results comparable to a manually tuned baseline from earlier work [32].

The urgency of efficient alarming systems for wildfires led to a need for models that can reliably predict burned areas. However, engineering and optimizing of these models is labour-intensive and requires expertise, making it inaccessible for many regions. AutoML offers potential solutions in various domains. However, it remains unclear to what extent this potential extends to burned area detection specifically.

3 Data

This research uses two datasets containing Earth Observation (EO) data. EO data is collected by satellites or aircrafts, observing the Earth's surface from above. Before elaborating on the specifications of the dataset, it is important to note that this data originates from the European Space Agency's (ESA) Sentinel-2 mission. This mission carries two identical satellites equipped with high-resolution Multispectral Instruments (MSI). The imagers feature 13 spectral bands, with wavelengths ranging from 443 nm to 2190 nm, covering the visible spectrum as well as Near-Infrared (NIR) and Shortwave-Infrared (SWIR) spectrums. The infrared bands are critical for this research, as thermal anomalies such as wildfires emit radiation in these specific wavelengths. The instrument covers a swath width of 250 kilometres and varying spatial resolutions of 10, 20 and 60 meters, depending on the spectral band.

3.1 SegTHRawS

This research uses the segmentation dataset SegTHRawS (Segmented Thermal Hotspots in Raw Sentinel-2 data). The data covers temperature hotspots like wildfires and volcano eruptions, accumulated over the whole globe. The dataset contains 100 samples, each corresponding to one Sentinel-2 image, and serves segmentation purposes. Each sample is cropped into multiple overlapping patches of 256x256 pixels (25% overlap in each direction, which avoids missing events on borders of patches), resulting in 700 patches [29]. Each pixel is annotated to one of three labels: no event, potential event and certain event, corresponding to the values 0.0, 0.5 and 1.0 respectively. The dataset uses three spectral bands from Sentinel-2: B8A (NIR), B11 (SWIR) and B12 (SWIR). They are useful in detecting thermal anomalies, since burned areas reflect in these wavelengths, while healthy vegetation does not. The band selection is based on the Normalized Burn Ratio, a well-known index for burned area detection. It is based on NIR and SWIR [31]. Chapter 5 elaborates on the application of this dataset.

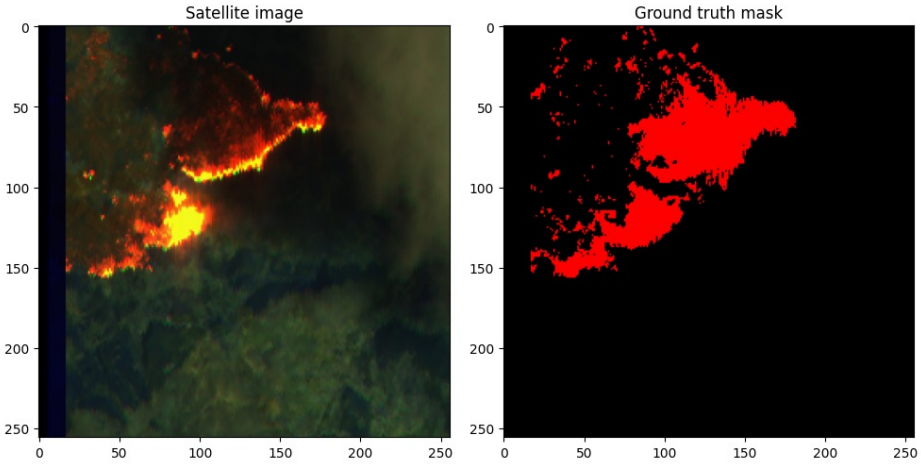

Figure 1: Example of a SegTHRawS patch, showing the satellite image (left) and the ground truth mask (right), where red pixels indicate certain fire events, orange pixels indicate potential events and black pixels indicate no event

3.2 OEOBench Burnt Area

This thesis also uses the semantic segmentation dataset OEOBench Burnt Area [10]. This dataset contains simulated images which cover 115 wildfire events across six continents and three fire types (boreal, temperate and tropical fires). The simulated images are derived from Sentinel-2 images, resampled and degraded to fit the sensor characteristics of the Φ-sat-2 satellite, resulting in 7784 patches of 256x256 pixels. Each pixel is annotated with one of four classes: background, burned area, clouds and water bodies. It uses seven spectral bands derived from Sentinel-2 imagery (B02, B03, B04, B05, B06, B07, B08). Unlike SegTHRawS, this dataset does not contain a SWIR band. In the absence of this band, the Normalized Burn Ratio cannot be used as a feature to train the models on.

4 Models and Used Methods

This chapter provides the theoretical background of the machine learning models used in this study. The principles and concepts of all models are introduced. The specific configurations, application and choices within this study are discussed in Chapter 5.

4.1 Majority Class

The Majority Class is the most fundamental baseline in this study. It predicts the most frequent class in the data for all instances, without any learning mechanism. It assigns each instance identical labels. Although this baseline is very simplistic, it reveals important information about more complex models. If a model can not outperform this baseline, it is implied that this model did not learn anything meaningful from the data. The classifier sets an expected performance level for other models.

In imbalanced datasets, the majority class classifier typically achieves high accuracy by predicting the dominant class for all instances, but it will have poor identification of the minority class. In the case of burned area datasets, there is a lot of imbalance in the data. Most of the time, the non-events or background classes will appear more frequently, since wildfires are rare. This makes for an informative lower baseline. A model that achieves on majority class level fails to detect any fire.

The specific configuration of the classifier and the motivation for the setting is discussed further in Chapter 5.

4.2 Random Forest Model

Random Forest is an ensemble classifier that uses decision trees to classify data. It produces multiple decision trees consisting of randomly selected subsets of training data [2]. It combines trees to produce more accurate and robust predictions than individual trees. This classifier gained popularity in remote sensing due to the accuracy of its predictions.

Within the individual decision trees, the Gini impurity is used to measure the quality of a split, and decide which split is the best. The Gini impurity quantifies the probability that a sample from a node would be correctly classified if its label were randomly assigned according to the class distribution of that node. For a node containing samples for K classes, the Gini impurity is defined as:

$$G = 1 - \sum_{k=1}^{K} (p_k^2) \tag{1}$$

where p_k is defined as the proportion of samples belonging to class k in that node. A Gini impurity of 0 would indicate a perfectly pure node with all samples belonging to the same class. At each split, the feature and threshold that minimise the weighted Gini impurity across the resulting child nodes are selected.

Decision trees are intuitive and interpretable, but they are also prone to overfitting [7]. A tree grown without constraints will perfectly fit the training data through memorization, but generalizes poorly to new data. To overcome overfitting, bagging (Bootstrap Aggregating) can be applied [6]. When using this method, each tree is trained on its own bootstrap sample. These samples differ slightly, by drawing data points with replacement, meaning some instances are repeated, while

others are left out. The combination of independently trained trees reduces variance, because mistakes made by individual trees are averaged out. This decreases the chance of a tree memorizing specific data and therefore reduces overfitting.

The Random Forest model extends the bagging method by randomly selecting subsets of features to consider at each split. This results in splits based on different, less dominant features, which reduces correlation between trees, making the ensemble more effective. The final prediction is based on combining the different trees.

The specific configuration for Random Forest used in this study is discussed further in Chapter 5.

4.3 XGBoost Model

XGBoost, or extreme Gradient Boosting [3], is a supervised ensemble learning method, which combines multiple weak decision trees into a single strong predictive model. XGBoost uses a method similar to bagging, namely boosting [8], which also combines multiple decision trees into one model. There are some differences however. While bagging trains its trees independently before combining their predictions, boosting combines weak trees, i.e., trees that only perform slightly better than just guessing. Boosting trains its trees sequentially. Each new tree corrects mistakes made by the previous trees. This correction process turns weak trees into a strong predictive model after multiple iterations. This method reduces bias rather than variance.

The way in which mistakes are passed on to new trees defines the model. XGBoost uses a method called gradient boosting. It trains trees based on residuals, the difference between the current predictions and the ground truth. By fitting new trees to these residuals, the model iteratively minimizes its overall error and moves the predictions towards the true values. To prevent overfitting caused by these iterative corrections, XGBoost applies built-in regularization. In addition to penalizing the number of leaves, XGBoost also penalizes the size of the prediction value assigned to each leaf (the leaf weight). Large leaf weights indicate that the tree is closely adjusting to a small number of specific examples rather than a general pattern, particularly for leaves with few data points. By penalizing the sum of squared leaf weights, the model discourages such large, overly specific corrections. Additionally, XGBoost normally grows trees until a maximum depth, and then applies backward pruning [8]. If the gain appears inefficient, the branch is removed. This method ensures that potential valuable splits deeper in a tree are not rejected too early.

The specific configuration for XGBoost used in this study is discussed further in Chapter 5.

4.4 U-Net Model

The U-Net model is a conventional neural network (CNN) designed for image segmentation [25]. Unlike many CNNs that produce one label per image for classification tasks, U-Net generates predictions on a pixel-based level. This preserves both local details and global context. This has proven especially effective in remote sensing, since it is more important to know *where* there is burned area and the shape, rather than *if* there is burned area. The architecture [25] consists of a contracting path, the encoder, and an expansive path, the decoder. The encoder extracts features through repeated convolutions, together with ReLU activation functions and max pooling. It learns increasingly abstract and global patterns, while halving the spatial resolution. The deepest point of the U-Net is the bottleneck. The representation of the data is the most abstract in this point. The decoder builds the spatial resolution up again, after passing the bottleneck. The

skip connections distinguish the U-net model from other CNNs. These connections rebuild the resolution, by integrating the high-resolution feature maps from the encoder directly with the features in the decoder. This enables the network to restore precise local details that were lost during downsampling. Lastly, the output layer reduces the feature maps given by the decoder and skip connections to one channel, while the dimensions of the input are maintained. The value per pixel is compressed between 0 and 1 by the sigmoid function and represents the probability of belonging to the fire class.

Loss function

U-Net uses a loss function to calculate the difference between the current predicted mask, and the ground truth mask. This difference is used to improve the predictions, by trying to minimize the loss. Since wildfire are exceptionally rare on a pixel basis, the datasets shows severe class imbalance. A standard Binary Cross-Entropy loss [14] treats all pixels equally, which means the background pixels dominate the loss, and the model learns to ignore the minority fire class. Focal Loss addresses this by down-weighting easy, correctly classified examples to reduce the impact of correct predictions and forcing the model to focuses more on incorrect predictions. The Binary Focal Loss function is as follows:

$$FL(p_i) = -\alpha_c(1 - p_i)^\gamma \cdot \log p_i \tag{2}$$

where p_i denotes the predicted probability for each pixel, a_c controls the weight of the event-class and γ determines the impact of correct predictions. A higher γ results in more focus on the minority class. The specific configuration of U-Net in this study is discussed further in Chapter 5.

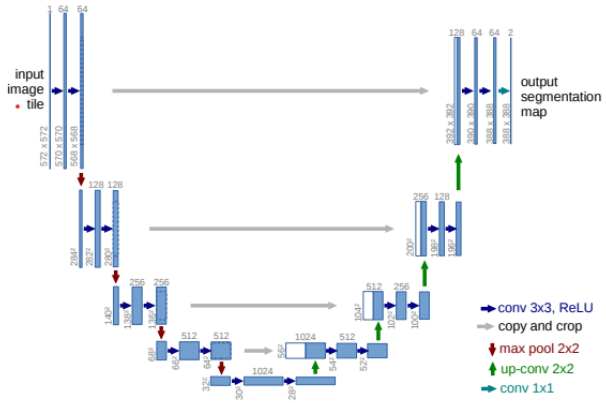

Figure 2: Example of a U-Net model, showing the expanding and contracting path, the bottleneck and the skip connections [25]

4.5 AutoML

In this study, an automated machine learning (AutoML) approach is evaluated alongside the baseline models. AutoML automates the process of searching for optimal hyperparameters and architectures [1, 33]. This eliminates a labour-intensive manual process, which often forms a bottleneck in research. While AutoML libraries [27] come in multiple forms ranging in customizability,

they must support the specific requirements of the task. For instance, AutoKeras provides a simplified interface, but currently does not support semantic segmentation.

To accommodate semantic segmentation, a balance between flexibility and simplicity is needed. KerasTuner [23] meets these requirements, as shown in Figure 3. KerasTuner has no predefined tasks, but rather adapts to the tasks given to it. This is the reason that it tunes hyperparameters, and not architectures. In this case, the chosen architecture is a U-Net model, and KerasTuner optimizes its hyperparameters.

This approach reduces the manual tuning effort while maintaining control over the model's structural design. This research uses Bayesian Optimization, one of the search algorithms that is available in KerasTuner [23]. This optimization uses a probabilistic model to select the hyperparameters for a new trial based on the previous trial. This results in an efficient search of hyperparameters. The specific configuration of KerasTuner used in this study is discussed further in Chapter 5.

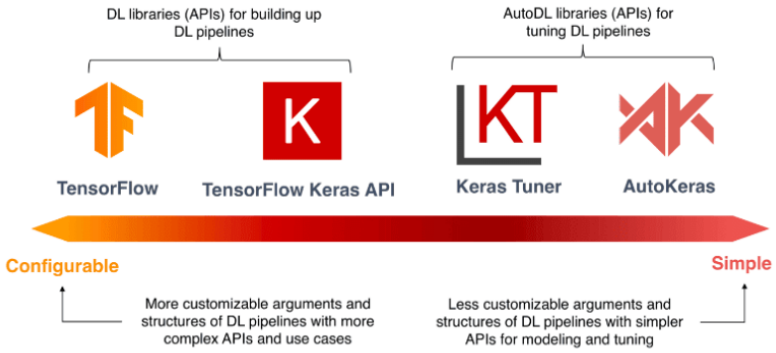

Figure 3: Spectrum of deep learning and AutoML libraries, illustrating the trade-off between configurability and simplicity. (Image adapted from [27])

5 Experimental Setup

This chapter addresses the methods this research is based on. First, the experimental design for each dataset is discussed, which will go into the cutoff and threshold analysis. Then, the evaluation metrics will be discussed, after which the pipelines of the baseline, deep-learning and AutoML approaches are elaborated further. The pipeline for the models differs slightly, so they are covered separately. One universal part of each pipeline is the averaging of results across seeds. For the SegTHRawS dataset, 10 seeds are applied, for the OEOBench Burnt Area dataset, there are 5 seeds. [1]

5.1 Experimental design

SegTHRawS

In this study, the data from SegTHRawS is converted to a new ground truth. This is done by defining cutoffs. The original ground truth consists of three levels of certainty about an event; no event, potential event and certain event, respectively represented by the labels 0.0, 0.5 and 1.0. The strictness of the fire definition in the ground truth affects which pixels are treated as fire during training. In order to evaluate the models' sensitivity to this definition, the ground truth is redefined to binary labels (no fire/fire) using four cutoff levels, as shown in Table 1. Figure 4 illustrates how the ground truth labels are assigned to the fire and no fire class based on cutoff level. For each cutoff level, fewer pixels belong to the fire class.

Table 1: Overview of the four cutoff definitions applied to SegTHRawS, where `ge` denotes greater than or equal to

Cutoff definitions	**Meaning**	**Number of datapoints**
`ge0` **(>= 0.0)** All pixels are fire	All ground truth labels >= 0.0 are redefined as fire	1127743488
`ge05` **(>= 0.5)**	All ground truth labels >= 0.5 are redefined as fire	355134
`ge10` **(>= 1.0)**	All ground truth labels >= 1.0 are redefined as fire	258248
`ge15` **(>= 1.5)** No pixels are fire	All ground truth labels >= 1.5 are redefined as fire	0

The models are trained on these binary variants of the dataset. Thresholds determine at which predicted probability a pixel is classified as fire. For all metrics, the threshold has a value of 0.5. For AUROC, the threshold ranges from 0.05 to 0.95, as described in 5.2.

Furthermore, to assess the robustness of the models for different train/test splits, each model is retrained and re-evaluated on 10 different 80/20 train/test splits, generated by 10 different random seeds. The seeds only affect the splits, and model hyperparameters or internal randomness (e.g. Random Forest's bootstrap sampling) are kept fixed. This isolates the effect of different train/test splits. Reported results are the mean and standard deviation across these 10 seeds.

[1] The implementation of the following method is available as code on the following GitHub page [12].

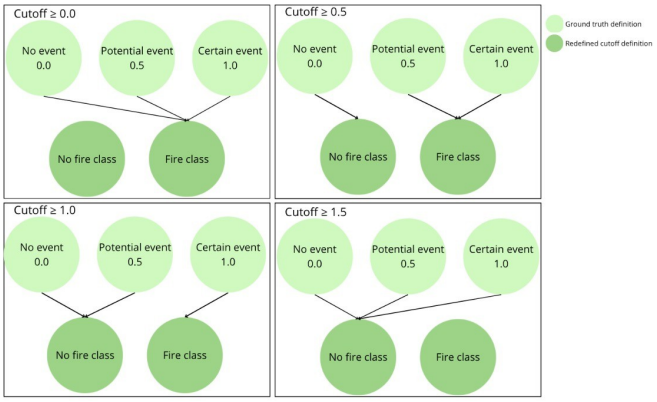

Figure 4: Representation of the ground truth labels assigned to new classes based on cutoff levels

OEOBench Burnt Area

The OEOBench Burnt Area dataset does not hold certainty levels, and therefore the new cutoff definitions described in Section 5.1 are not applied to this dataset. Instead, the four classes are binarized directly: pixels labelled as burned area are treated as the fire event class, and all other pixels (background, cloud, water) are treated as the no event class. It follows a same approach as the class merging applied on SegTHRawS at threshold 0.5. This ensures a consistent binary formulation of the labels across the datasets and a comparison of mIoU scores across the datasets. The OEOBench Burnt Area dataset is considerably more balanced than the SegTHRawS dataset. Approximately 18.9% of pixels belong to the fire class. This proportion is maintained across the pipeline.

The OEOBench Burnt Area dataset is also submitted to different train/test splits to asses the robustness of the predictions. For this dataset, because of time limitations, the models are retrained on 3 different random seeds, resulting in different train/test splits. Again, these seeds affect only the split, and not internal architecture settings or randomness.

5.2 Evaluation metrics

The primary metric to evaluate the model's performance is the mIoU (mean intersection over union). This metric measures the overlap of pixels in the fire class between the ground truth mask GT and the predicted mask P.

$$IoU = \frac{|GT \cap P|}{|GT \cup P|} \tag{3}$$

The IoU score is 1 for pixel with a correctly predicted fire label, and 0 otherwise. The mIoU is the average IoU across all patches. Figure 5 illustrates the four possible combinations of ground truth masks and predicted masks, and their corresponding mIoU score.

Only when both the ground truth and the prediction mask indicate fire (class 1), the mIoU score equals 1. Notably, a correctly predicted background pixel (both masks indicate no fire) also scores 0, since the mIoU score only evaluates the fire class. A model could therefore predict perfectly, but score 0, when no fire is present.This behaviour is particularly relevant for the `no fire` cutoff. Since the segmentation task depends on correctly identifying the shape and extent of a burned

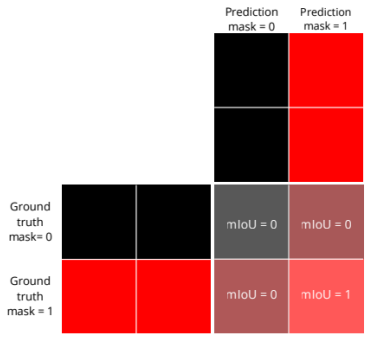

Figure 5: Example of mIoU score for different ground truth masks and prediction masks

area, rather than only pixel-level correctness, mIoU is the leading metric for comparing models' performance throughout the thesis and Chapter 6. In addition to the mIoU score, the AUROC (area under ROC curve) is reported as a secondary metric. The ROC curve shows the performance across classification thresholds and is robust to class imbalance. While the mIoU evaluates the spatial overlap at a fixed threshold, AUROC evaluates performance across all thresholds. The ROC curve, and the corresponding AUROC score, show how much a model has learned compared to a random classifier. An AUROC score of 0.5 corresponds to random guessing.

Accuracy, precision, recall and F1-scores are reported in Appendix A as supporting metrics. Accuracy measures the proportion of correct predictions. However, it can be misleading and unreliable in imbalanced datasets as a model predicting only the majority class still score highly. Precision and recall capture the trade-off between false alarms and missed events, and the F1-score combines the two into a balanced metric.

5.3 Baseline pipeline

SegTHRawS

Three baseline models serve as performance benchmarks for comparison to the optimized model by KerasTuner. Therefore, the models are trained and evaluated. The baseline models are trained on input consisting of 17208 patches of 256x256 pixels per patch with three spectral bands (NIR and SWIR), as discussed in Chapter 3. This includes 824 event patches, 2894 potential event patches and 13490 no event patches. These patches are extracted using the same 256x256 patch size as describe by Traba et al., but with a larger set of no-event patches added, to provide sufficient background examples for training. All baseline models (Majority Class, Random Forest and XGBoost) are trained on SegTHRawS using a 80/20 train/test split at the patch level, before the patches are flattened to pixel-level samples. For fair comparison with both U-Net models, which are evaluated on the complete test set, the three baselines are trained and evaluated on a large-scale pixel sample: a stratified random subsample of 100 million pixels is drawn from the full pixel-level dataset, both for training and testing. This subsample size was chosen as a practical upper bound, since the full flattened dataset was too large to train the models within reasonable computational bounds. The sampled pixels are stratified on the binary fire/no-fire labels, and therefore preserve class ratio. At the evaluation cutoff `ge05`, the fire-class consists of approximately 40000 pixels.

The cutoff definitions `all fire` and `no fire`, result in edge cases where the dataset contains

only one class. The baseline models require at least two classes and therefore fail to train on these trivial cases. To maintain a consisted pipeline and to evaluate the edge cases, the models are substituted with a constant-prediction baseline model (DummyClassifier) for these specific cutoffs. Table 2 shows the important settings of the models and their respective motivation.

Where possible, standard recommended values were used, as the purpose of the models was to establish a performance benchmark without hyperparameter optimization. The effect of optimization can then be evaluated more accurately, after comparing the baselines to KerasTuner's results.

Table 2: Overview of model architecture details and their motivations for the baseline models

Model	Hyperparameter	Value	Motivation
Majority Class Classifier	strategy	prior	Each prediction gets a probability equal to the proportion of fire pixels in the training data. This way, predictions can be analysed over different thresholds.
Random Forest	n_estimators	50	Sufficient for stable predictions with limited computational costs
	max_depth	15	Tree depth that prevents overfitting but allows sufficient complexity
	class_weight	balanced	To deal with imbalance in datasets
XGBoost	n_estimators	50	Standard recommended value
	max_depth	6	Standard recommended value in documentation [8]
	learning_rate	0.1	Selected from recommended value in documentation [8]
	scale_pos_weight	ratio class 0 to class 1	To deal with imbalance in datasets

OEOBench Burnt Area

The same baseline models are trained on the OEOBench Burnt Area dataset. The same architecture configuration as for SegTHRawS is used, as shown in Table 2. Since this dataset does not require retraining on different cutoff definitions, each model is trained once per seed, on the binarized labels. The same pixel-level subsampling approach is applied, using the same cap of 100 million pixels as for SegTHRawS. For this dataset, only the training set exceeds this cap, and is therefore subsampled. The full test set is used, as it falls below the pixel cap.

5.4 Deep learning pipeline

SegTHRawS

The U-Net model is trained on the same input as the baseline models, over 17000 patches of 256x256 pixels with three spectral bands and an 80/20 train/test split. This allows for direct comparisons. The U-Net is implemented using TensorFlow 2.12.0 and Keras 2.12.0. Table 3 provides an overview of the model configuration and the motivation behind each choice.

Where possible, standard values were used, as the purpose of the U-Net model is to provide a deep learning benchmark without automated hyperparameter optimization. The effect of this optimization can then be evaluated more accurately, after comparing to KerasTuner's results.

Although mIoU is the primary metric used to evaluate segmentation quality in this thesis, the model is trained using Binary Focal Crossentropy rather than an IoU-based loss. It reflects common practice in segmentation research, where models are typically trained using cross-entropy-based loss, despite it resulting in misalignment between the training objective and evaluation metric [17]. In its standard form, mIoU relies on a hard threshold for classification of pixels, which makes it non-differentiable and unsuitable for gradient-based optimization. Implementation of a differentiable approximation of IoU requires replacing the hard threshold with the model's raw probability output and using the metric $1 - mIoU$ as the loss to be minimized during training [24].

Table 3: Overview of model architecture details and their motivations for U-Net pipeline

Architecture component		**Value**	**Motivation**
Filters	Encoder	16, 32	Compact architecture for limited amount of data
	Bottleneck	64	
	Decoder	32, 16	
Activation function	Encoder/ decoder	ReLU	Standard function, as discussed in Section 4.4
	Output	Sigmoid	Standard for output between 0 and 1, as discussed in Section 4.4
Epochs		50	Chosen to allow sufficient iterations for convergence, within limits of computational resources.
Batch size		32	Standard value for GPU training
Optimizer		Adam	Widley used for deep learning tasks [16]
Loss function		BinaryFocalCrossentropy (alpha=0.9, gamma=2.0)	To deal with class imbalance, as discussed in Section 4.4. Alpha and gamma put more weight on the minority class to counterbalance the imbalanced dataset.

OEOBench Burnt Area

The U-Net model trained on the OEOBench Burnt Area has the same 80/20 split and architecture details as for the SegTHRawS dataset in Table 3, but the architecture is adjusted to cover 7 spectral bands instead of 3 for SegTHRawS. Furthermore, the model is trained on a subset of 300 patches, instead of the available 7784, for a feasible training time and memory.

5.5 KerasTuner pipeline

SegTHRawS

KerasTuner (v1.4.8) is used for hyperparameter optimization of the U-Net architecture. To be able to compare results, KerasTuner also used the same 17000 patches of 256x256 pixels with three spectral bands, with an 80/20 train/test split. The architecture details and their motivations are summarised in Table 4.

To ensure a fair comparison with the untuned U-Net model, several settings are fixed and not tuned. These configurations are the optimizer, the loss function, the batch size and number of epochs. This isolates the effect of the hyperparameters that are tuned by KerasTuner, for optimal comparison.

An important characteristic of the design of the tuned model is the calculation of its capacity. The number of filters per encoder or decoder level is not a tunable parameter. They are calculated

on the chosen `max_level_size` and `num_levels`, where the number of filters doubles each step up the bottleneck. The max level size search space is restricted to powers of two (32, 64, 128), which is consistent with common CNN design practices.

Furthermore, the search space includes a dropout rate as a variable parameter. The dropout rate randomly deactivates fractions of neurons during training [28]. The U-Net is forced to learn more robust and generalized features, ensuring regularization and countering overfitting. The dropout rate is searched steps of 0.1, instead of continuously, so that it can be explored within a limited number of trials adequately. The dropout search space, between 0.0 and 0.5 with steps of 0.1, is based on common practices in other hyperparameter search studies [22].

Lastly, the KerasTuner optimizes the learning rate on a logarithmic scale, between 10^{-5} and 10^{-2} [4]. The learning rate determines the step size that the optimizer can take when updating weights in the model. A higher learning rate could lead a model to miss the optimal combination of parameters, while a low learning rate could result in a very slow training process. The logarithmic scale ensures that multiple orders of magnitude are explored to balance too high and too low learning rates.

Although mIoU is the primary metric for the evaluation of segmentation quality in this thesis, the model is still trained using Binary Focal Crossentropy as its loss function, for the reasons discussed in Section 5.4. This limitation does not extend to KerasTuner's search objective. An objective is used to compare completed trials and therefore does not have to be differentiable. KerasTuner optimized directly for validation mIoU, which ensures that the search space chooses the hyperparameters based on the metric that is used for the evaluation of the models in this thesis.

Table 4: Overview of the tunable hyperparameters and their motivations in the KerasTuner pipeline

Hyperparameter	**Search space (values)**	**Motivation**
Number of levels	2, 3, 4	Standard number of levels to test for optimization
Max level size	32, 64, 128	To test both smaller and bigger models than the baseline U-Net
Learning rate	Float: 10^{-5} to 10^{-2} (logarithmic scale)	Standard learning rates to test for optimization
Dropout rate	Float: 0.0 to 0.5 (step 0.1)	Standard dropout rate to test for optimization
Batch size	32	Equal to baseline U-Net for comparative reasons
Epochs	50	Equal to baseline U-Net for comparative reasons
Optimizer	Adam	Equal to baseline U-Net for comparative reasons
Loss function	BinaryFocalCrossentropy (alpha=0.9, gamma=2.0)	Equal to baseline U-Net for comparative reasons

OEOBench Burnt Area

The same hyperparameter optimization, including the mIoU search objective, is performed on this dataset, with the same search space as for SegTHRawS in Table 4. It uses the same 300 patches as the untuned U-Net model does.

6 Results

In this chapter the results of the two datasets are presented. First, the results for SegTHRawS of the baseline models (Majority class, Random Forest and XGBoost) in Section 6.1, the untuned deep-learning model (U-Net), in Section 6.2, and the AutoML model (KerasTuner), in Section 6.3, are presented. A comparison will show the model's performance relatively, in Section 6.4, to show the effect of automated hyperparameter optimization. After this comparison, results for the OEOBench Burnt Area datasets are shown in Section 6.5. In A, additional metrics for the SegTHRawS data are shown in Table 8 and for OEOBench Burnt Area in Table 9. And the classification reports for SegTHRawS are shown in 10.

6.1 Baseline results

The results of the baseline models are shown below. In Table 5, the mIoU score for each model is shown. The results are across the different cutoff definitions and averaged across 10 different random seeds. It shows both the mean and the standard deviation. In Figure 6, the ROC curve for these models is presented along with the other models, also across 10 seeds.

Table 5: mIoU scores of the baseline models at threshold 0.5 across all cutoff definitions, averaged over 10 seeds (showing mean ± standard deviation)

Cutoff	Majority Class	Random Forest	XGBoost
`all fire`	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
`ge05`	0.00 ± 0.00	0.14 ± 0.03	0.08 ± 0.05
`ge10`	0.00 ± 0.00	0.17 ± 0.04	0.09 ± 0.09
`no fire`	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00

It needs to be emphasized that the baseline models were not trained on all pixels of the dataset for memory purposes. This differs from the training process of both U-Net models. The baseline models are trained on 100 million pixels, of which approximately 40000 pixels belong in the fire class on the `ge05` cutoff definition.

The Majority class classifier shows an expected prediction outcome for the imbalanced data. It predicts the majority class and scores 1.00 ± 0.00 for the `all fire` cutoff. For the other classes, with the fire class as a minority, it score 0.00 ± 0.00. The ROC curve of the majority class lies directly on top of the random guess line, with an AUROC score of 0.50 ± 0.00, which equals that of a random classifier.

The Random Forest model scores 1.00 ± 0.00 as well for the trivial `all fire` cutoff. For the two non-trivial cutoffs, it scores 0.14 ± 0.03 and 0.17 ± 0.04. It scores higher for the stricter cutoff. For the `no fire` cutoff, it scores 0.00 ± 0.00. The ROC curve of Random Forest lies well above the Majority class classifier and the random classifier, with AUROC scores of 0.99 ± 0.04 for `ge05` and 1.00 ± 0.01 for `ge10`.

XGBoost achieves a score of 0.08 ± 0.05 on the `ge05` cutoff definition and 0.09 ± 0.09 on the `ge10` definition, again with the score increasing on the stricter cutoff. For the `all fire` and `no fire` definitions it scores 1.00 ± 0.00 and 0.00 ± 0.00 respectively. The ROC curve of XGBoost lies above the Random Forest curve with scores of 0.99 ± 0.02 and 1.00 ± 0.00 for the non-trivial cutoffs, when averaged across 10 seeds.

The results of the baseline models show an expected behaviour of the Majority Class classifier. Random Forest and XGBoost perform better than the Majority Class classifier, with Random Forest performing better than XGBoost. A pattern that both models show is an increase in performance when moving from `ge05` to `ge10`.

6.2 Deep learning results

The results of the U-Net model are shown below. In Table 6, the metrics per cutoff across 10 seeds are shown, also showing both the mean and standard deviation across the seeds. In Figure 6, the ROC curve is presented, along with the other models. Unlike the baseline models, the U-Net model was trained on all available pixels in the dataset, containing over 225 million pixels, of which 90529 are in the fire class and 225484383 are in the no fire class.

Table 6: mIoU scores of the U-Net model at threshold 0.5 across all cutoff definitions, averaged over 10 seeds (showing mean ± standard deviation)

Cutoff	**U-Net**
`all fire`	1.00 ± 0.00
`ge05`	0.52 ± 0.06
`ge10`	0.62 ± 0.05
`no fire`	0.00 ± 0.00

The untuned U-Net model scores 1.00 ± 0.00 on the `all fire` cutoff definition and 0.00 ± 0.00 on the `no fire` definition. For the two non-trivial definitions, it scores 0.52 ± 0.06 on the `ge05` cutoff and 0.62 ± 0.05 on the `ge10` cutoff. Again, the scores increase on the stricter cutoff, following the pattern of the baseline models. The ROC curve lies above all other baseline models, scoring 1.00 ± 0.00 for both non-trivial definitions.

6.3 KerasTuner results

The results of the tuned U-Net model are shown below. In Table 7 the metrics per cutoff are shown across 10 seeds. In Figure 6, the ROC curve is presented together with the other models. Like the U-Net model, this model is trained on all available pixels.

Table 7: mIoU scores of the tuned U-Net (KerasTuner) model at threshold 0.5 across all cutoff definitions, averaged over 10 seeds (showing mean ± standard deviation)

Cutoff	**U-Net (KerasTuner)**
`all fire`	1.00 ± 0.00
`ge05`	0.54 ± 0.05
`ge10`	0.64 ± 0.05
`no fire`	0.00 ± 0.00

The model achieves scores of 0.54 ± 0.05 for the `ge05` cutoff definition and 0.64 ± 0.05 for the `ge10` definition, and the trivial cutoff definitions `all fire` and `no fire` score 1.00 ± 0.00 and

0.00 ± 0.00, respectively. The ROC curve is identical to the manually tuned U-Net, with an AUROC score of 1.00 ± 0.00 for both non-trivial cutoff definitions as well.

6.4 Comparison

Figure 6 and Figure 7 show a comparison of all models in an ROC curve and an mIoU graph across the different cutoff definitions, again showing the means and standard deviations of multiple train/test splits. Figure 8 shows the boxplots for all models of the mIoU score for the `ge05` and `ge10` cutoffs, since the edge cases show the trivial values (1.00 or 0.00) with a standard deviation of 0.00. These graphs make an straightforward direct comparison between the models possible.

Figure 6 shows that U-Net and KerasTuner have higher performance than the other models. All models score well above a random classifier. The U-Net model and the tuned U-Net model lie together, the XGBoost scores lower and the Random Forest scoring somewhat lower still. The Majority Class scores 0.50, the same as a random classifier, which is to be expected.

Figure 7 shows the mIoU scores across different cutoff definitions. The trend that was visible in Sections 6.1, 6.2 and 6.3 is also clearly visible here; when moving from `ge05` to `ge10`, almost all scores rise slightly. The two types of models lie close together; Random Forest and XGBoost (`ge05`: $0.14 - 0.08$ and `ge10`: $0.17 - 0.09$), and the U-Net and the tuned U-Net (`ge05`: $0.52 - 0.54$ and `ge10`: $0.62 - 0.64$). KerasTuner scores the highest, with a score of 0.64 on the `ge10` cutoff.

Figure 8 shows the distribution of the mIoU scores across multiple train/test splits and illustrates the stability and variance of the models. It is clear that both U-Net models are more stable, for both cutoffs, than the baseline models, with tight boxes and few outliers. Among the baselines, XGBoost shows a wider spread than Random Forest on both cutoffs, despite an including an outlier on the `ge10` cutoff. Moreover, the tuned model by KerasTuner shows a smaller distribution than the manually tuned U-Net model, most clearly visible on the `ge10` cutoff.

6.5 OEOBench Burnt Area

The results of this dataset show the robustness of the effectiveness of KerasTuner across datasets. Therefore, the boxplots covering mIoU scores in Figure 9 are shown. The Majority Class scores 0.00 reflecting the class imbalance. Random Forest scores 0.73 and XGBoost scores 0.72, with tight distributions.The untuned U-Net achieves a score of 0.70, which is slightly lower than the other baseline models, and shows a wider distribution. The tuned U-Net model by KerasTuner achieves the highest median score of 0.80 and a tight distribution. Compared to Figure 8, the tuned U-Net achieves the highest score again, with a narrow spread. Unlike SegTHRawS, Random Forest and XGBoost score higher than the untuned U-Net model and show a tighter spread.

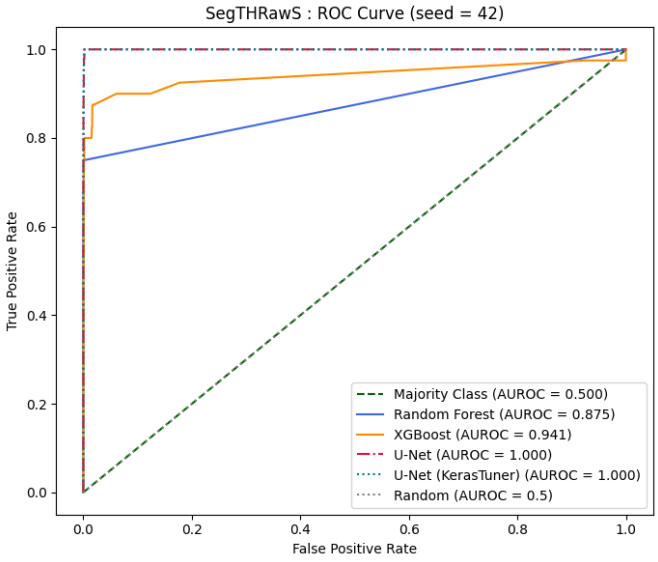

Figure 6: ROC curves for all models at threshold 0.5, on fixed seed (`seed` = 42), for the SegTHRawS data

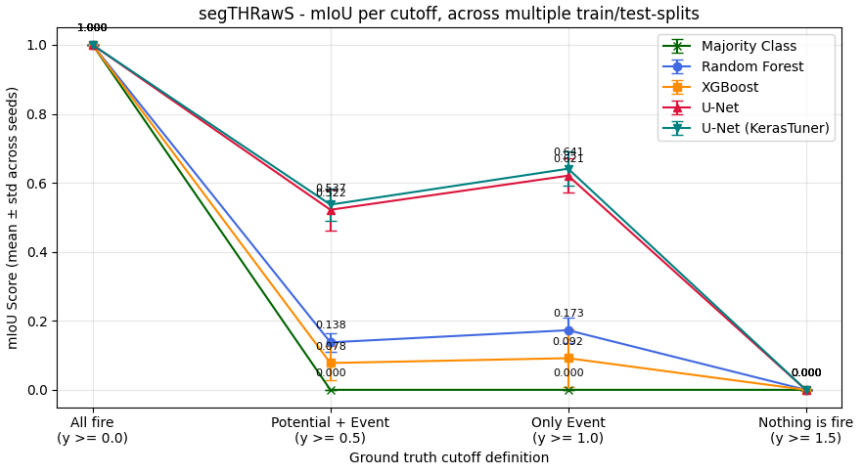

Figure 7: mIoU scores for all models over the different cutoff definitions across 10 different seeds, for the SegTHRawS data

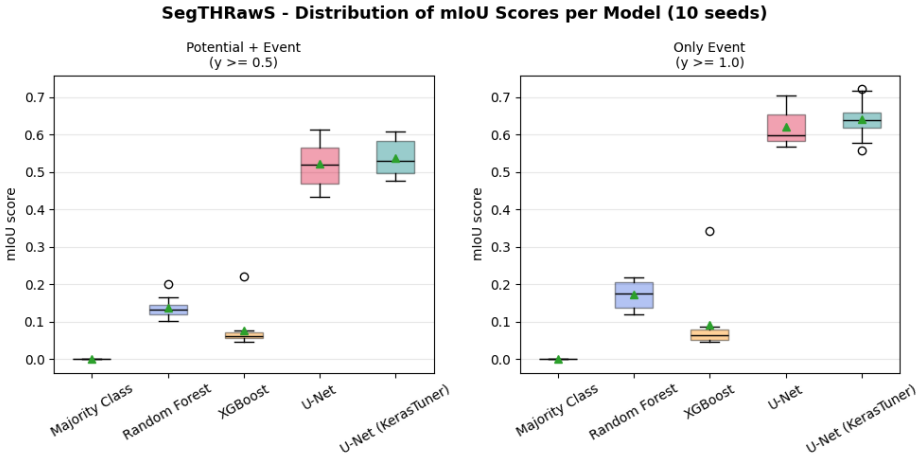

Figure 8: Boxplots of mIoU scores for all models over the different cutoff definitions across 10 different seeds, for the SegTRHawS data

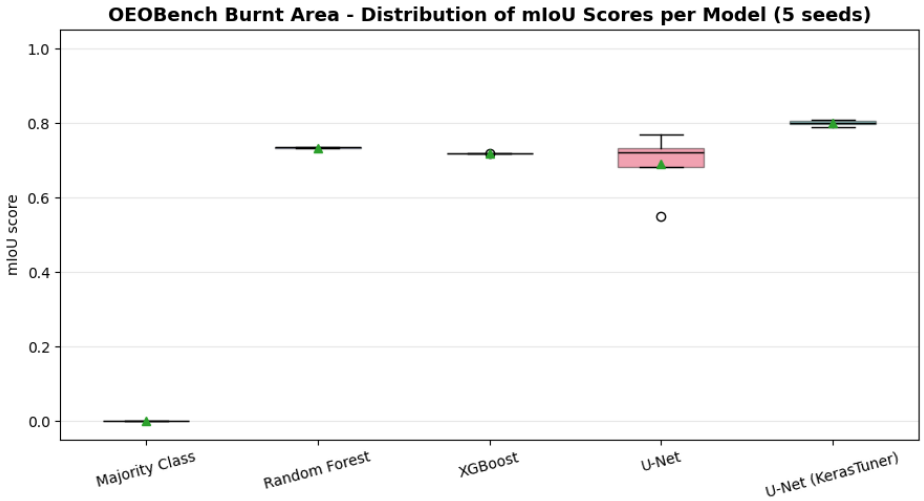

Figure 9: Boxplots of mIoU scores for all models across 5 different seeds, for the OEOBench Burnt Area data

7 Discussion

This thesis compared the performance levels between baseline models, a deep learning model and a model tuned by KerasTuner, to analyse the effect of AutoML for burned area detection. Results indicate a performance of the AutoML model at least as good as the untuned model, implying that there is potential for AutoML in this specific domain. The evaluation across multiple random seeds demonstrated that the AutoML approach achieves more robust and stable results than the other models. The results also show a consistent increase in performance when moving to stricter fire definitions. This would imply that certainty of annotation plays an important role in model performance. The results of the evaluation of the models across multiple seeds of the OEOBench Burnt Area dataset also show that KerasTuner achieves the highest median mIoU score with a tight distribution. This reinforces the effectiveness of KerasTuner as hyperparameter optimization technique for burned area detection across multiple datasets.

However, the limitations of this research should be taken into account. First, there are some notable shortcomings in the used data. This research only uses two datasets, so the results may not generalize well to other datasets. This limits the conclusion that can be drawn about the effectiveness of AutoML in this domain. Furthermore, the datasets only cover small amounts of data. Together with strong class-imbalance of the SegTHRawS data this also can have its effects on the results and their reliability. For the OEOBench Burnt Area dataset specifically, the limited amount of patches used to train both U-Net models, in combination with the smaller number of seeds compared to SegTHRawS, limits the statistical robustness of the results.

Then, there are limitations in the training of KerasTuner. Namely the training with relatively few trials, low number of epochs and a small set of tunable hyperparameters. More trials, epochs and variable hyperparameters would provide the model more search space, and possibly different results. This could potentially result in even performance scores higher scores for KerasTuner. Further research would provide more insight into the effectiveness of AutoML.

The final comparison also shows constraints. It only compares datasets with segmentation tasks, so generalization of the results is not possible across machine learning tasks. Furthermore, only one AutoML model was compared, so it could be possible that only this model shows potential. It therefore remains unclear if other AutoML models perform up to KerasTuner's potential, or possibly show better outcomes.

Lastly, while the current results show the stability and variance through the use of 10 random seeds, this number of seeds remains relatively limited. The results show whether the observed performance differences between models are consistent or coincidental.

Future research could expand in multiple directions. First, it could focus on the shortcomings by looking into more datasets, models and machine learning tasks, to provide broader comparisons and stronger conclusions about the general effect of AutoML for burned area detection. Training with more seeds and trials would allow for more robust comparisons, which would reduce the risk of coincidental results. Future research could also look into more specific regions and vegetation types, to investigate whether AutoML remains effective in more specific or different circumstances. This would make burned area detection more precise and applicable on smaller or different scales.

8 Conclusion

This thesis aimed to analyse to what extent automated machine learning can optimize binary segmentation-based detection of burned areas compared to baseline models. A comparison between three baseline models, an untuned deep learning model and a tuned deep learning model by KerasTuner gave insight into the effectiveness of hyperparameter optimization of AutoML.

From the results of the training of the SegTHRawS dataset, averaged over 10 random seeds, it can be concluded that hyperparameter optimization by KerasTuner can improve performance. Especially on the stricter annotations it can lead to better predictions: an absolute increase of 0.02, compared to the untuned U-Net model at `ge10`, and 0.29 compared to the best performing baseline model Random Forest on `ge10` for the mIoU score. The ROC curve of the tuned model also exceeds the scores of other models, corresponding with an AUROC score of 1.00.

The results suggest that the interpretation of the data annotation influences the performance across all models. This is visible in the stricter interpretation of what is classified as fire, and likely has further implications for data-centric methods. Among other things, it suggests that investing in annotation quality can also improve burned area detection performance, even without implementing AutoML.

The findings are supported by the evaluation of the OEOBench Burnt Area dataset. KerasTuner achieved an mIoU of0.2 higher than the untuned U-Net (`ge10` cutoff definition) and of 0.47 higher than Random Forest (`ge10` cutoff definition), which is the highest scoring baseline model. This suggest that the effects of hyperparameter optimization are not limited to one dataset, but show generalizability across datasets.

The effectiveness of hyperparameter tuning of deep learning models in burned area detection suggests a couple of things. First, manual tuning of models would be unnecessary, making the development more cost- and time-efficient, while maintaining or even improving performance. This also makes the models more accessible to regions with less machine learning resources. Secondly, improved burned area detection provides more accurate ground truth data, which can improve active wildfire prediction models. Together, this would result in better early response methods, and therefore might lessen the damage of wild fires.

In conclusion, automated machine learning shows clear potential for the optimization of burned area detection. As wildfires grow in frequency and severity, the need for efficient and accessible detection models has never been greater, and AuoML can offer a solution to the time and labour intensive problem of tuning detection models.

References

[1] Mitra Baratchi, Can Wang, Steffen Limmer, Jan N. van Rijn, Holger H. Hoos, Thomas Bäck, and Markus Olhofer. Automated machine learning: past, present and future. *Artificial Intelligence Review*, 57(5):122, 2024.

[2] Mariana Belgiu and Lucian Drăguţ. Random forest in remote sensing: A review of applications and future directions. *ISPRS journal of photogrammetry and remote sensing*, 114:24–31, 2016.

[3] Hoss Belyadi and Alireza Haghighat. Supervised learning. In *Machine Learning Guide for Oil and Gas Using Python*, chapter 5, page 169–295. Elsevier, 2021.

[4] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 437–478. Springer, 2012.

[5] Luigi Boschetti, David P. Roy, Christopher O. Justice, and Michael L. Humber. Modis–landsat fusion for large area 30m burned area mapping. *Remote Sensing of Environment*, 161:27–42, 2015.

[6] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794. ACM, 2016.

[9] Emilio Chuvieco. Historical background and current developments for mapping burned area from satellite earth observation. *Remote Sensing of Environment*, 225:45–64, 2019.

[10] Roberto Del Prete, Parampuneet Kaur Thind, Andrea Mazzeo, Matthew Whitley, Lorenzo Papa, Nicolas Longépé, and Gabriele Meoni. Optimizing deep learning models for on-orbit deployment through neural architecture search. *Scientific Reports*, 15(1):37783, 2025.

[11] S.H. Doerr, R.A. Shakesby, W.H. Blake, C.J. Chafer, G.S. Humphreys, and P.J. Wallbrink. Effects of differing wildfire severities on soil wettability and implications for hydrological response. *Journal of Hydrology*, 319(1):295–311, 2006.

[12] Laura Faas. Automated machine learning for wildfire detection. https://github.com/LauraCFaas/bscthesis, 2026.

[13] Nicholas R Goodwin and Lisa J Collett. Development of an automated method for mapping fire history captured in landsat tm and etm+ time series across queensland, australia. *Remote Sensing of Environment*, 148:206–221, 2014.

[14] Priya Goyal. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[15] Todd J. Hawbaker, Melanie K. Vanderhoof, Yen-Ju Beal, Joshua D. Takacs, Gail L. Schmidt, Jeff T. Falgout, Brad Williams, Nicole M. Fairaux, Megan K. Caldwell, Joshua J. Picotte, Stephen M. Howard, Susan Stitt, and John L. Dwyer. Mapping burned areas using dense time-series of landsat data. *Remote Sensing of Environment*, 198:504–522, 2017.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[17] Hao Li, Chenxin Tao, Xizhou Zhu, Xiaogang Wang, Gao Huang, and Jifeng Dai. Auto seg-loss: Searching metric surrogates for semantic segmentation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[18] James MacCarthy, Jessica Richter, Sasha Tyukavina, Mikaela Weisse, and Nancy Harris. The latest data confirms: Forest fires are getting worse, 2023.

[19] Christopher J Moran, Matthew P Thompson, Bryce A Young, Joe H Scott, and Melissa R Jaffe. Benchmarking performance of annual burn probability modeling against subsequent wildfire activity in california. *Scientific Reports*, 15(1):23699, 2025.

[20] World Health Organization. Wildfires, 2019. World Health Organization.

[21] World Health Organization. Climate change, 2023. World Health Organization.

[22] Atilla Özgür and Sevim Seda Yamaç. Modelling of daily reference evapotranspiration using deep neural network in different climates. *Journal of Applied Water Engineering and Research*, 14(2):219–236, 2026.

[23] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner, 2019.

[24] Md Atiqur Rahman and Yang Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In *International symposium on visual computing*, pages 234–244. Springer, 2016.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[26] Keisha Rukikaire. Number of wildfires to rise by 50 per cent by 2100 and governments are not prepared, experts warn, 2023. United Nations Environment Programme.

[27] Qingquan Song, Haifeng Jin, and Xia Hu. *Automated machine learning in action*. Simon and Schuster, 2022.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[29] Cristopher Castro Traba, David Rijlaarsdam, Jian Guo, Roberto Del Prete, and Gabriele Meoni. Towards onboard thermal hotspots segmentation with raw multispectral satellite imagery. *International Journal of Applied Earth Observation and Geoinformation*, 146:105095, 2026.

[30] Devis Tuia, Konrad Schindler, Begüm Demir, Xiao Xiang Zhu, Mrinalini Kochupillai, Sašo Džeroski, Jan N. van Rijn, Holger H. Hoos, Fabio Del Frate, Mihai Datcu, Volker Markl, Bertrand Le Saux, Rochelle Schneider, and Gustau Camps-Valls. Artificial intelligence to advance earth observation. 2024.

[31] United States Geological Survey. Landsat normalized burn ratio, 2024.

[32] Sven van Collenburg. Automated machine learning for sea ice concentration charting, 2024. BSc Thesis, Leiden University.

[33] Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers, 2023.

A Additional performance metrics

Table 8: Full performance metrics for all models at threshold 0.5 across all cutoff definitions, averaged over 10 seeds (showing mean ± standard deviation) for SegTHRawS

Majority Class	**Acc.**	**Prec.**	**Rec.**	**F1**	**mIoU**	**AUROC**
`all fire`	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	NaN
`ge05`	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.50 ± 0.00
`ge10`	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.50 ± 0.00
`no fire`	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	NaN
Random Forest	**Acc.**	**Prec.**	**Rec.**	**F1**	**mIoU**	**AUROC**
`all fire`	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	NaN
`ge05`	1.00 ± 0.00	0.22 ± 0.26	0.90 ± 0.24	0.24 ± 0.04	0.14 ± 0.03	0.99 ± 0.04
`ge10`	1.00 ± 0.00	0.25 ± 0.25	0.90 ± 0.23	0.29 ± 0.05	0.17 ± 0.04	1.00 ± 0.01
`no fire`	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	NaN
XGBoost	**Acc.**	**Prec.**	**Rec.**	**F1**	**mIoU**	**AUROC**
`all fire`	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	NaN
`ge05`	1.00 ± 0.00	0.08 ± 0.06	0.97 ± 0.09	0.14 ± 0.07	0.08 ± 0.05	0.99 ± 0.02
`ge10`	1.00 ± 0.00	0.10 ± 0.10	0.97 ± 0.08	0.16 ± 0.12	0.09 ± 0.09	1.00 ± 0.03
`no fire`	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	NaN
U-Net	**Acc.**	**Prec.**	**Rec.**	**F1**	**mIoU**	**AUROC**
`all fire`	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	NaN
`ge05`	0.99 ± 0.00	0.84 ± 0.02	0.58 ± 0.07	0.68 ± 0.05	0.52 ± 0.06	1.00 ± 0.00
`ge10`	1.00 ± 0.00	0.86 ± 0.05	0.69 ± 0.07	0.77 ± 0.04	0.62 ± 0.05	1.00 ± 0.00
`no fire`	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	NaN
U-Net (KerasTuner)	**Acc.**	**Prec.**	**Rec.**	**F1**	**mIoU**	**AUROC**
`all fire`	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	NaN
`ge05`	0.99 ± 0.00	0.79 ± 0.05	0.63 ± 0.05	0.70 ± 0.04	0.54 ± 0.05	1.00 ± 0.00
`ge10`	1.00 ± 0.00	0.89 ± 0.03	0.70 ± 0.05	0.78 ± 0.04	0.64 ± 0.05	1.00 ± 0.00
`no fire`	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	NaN

Table 9: Performance metrics showing accuracy, F1-score, mIoU and AUROC for all models at threshold 0.5, averaged over 10 seeds (showing mean ± standard deviation) for the OEOBench Burnt Area dataset

	Acc.	**F1**	**mIoU**	**AUROC**
Majority Class	0.81 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.50 ± 0.00
Random Forest	0.93 ± 0.00	0.85 ± 0.00	0.73 ± 0.00	0.98 ± 0.00
XGBoost	0.93 ± 0.00	0.84 ± 0.00	0.72 ± 0.00	0.98 ± 0.00
U-Net	0.84 ± 0.02	0.81 ± 0.06	0.69 ± 0.08	0.98 ± 0.01
U-Net (KerasTuner)	0.96 ± 0.02	0.89 ± 0.04	0.80 ± 0.07	0.99 ± 0.02

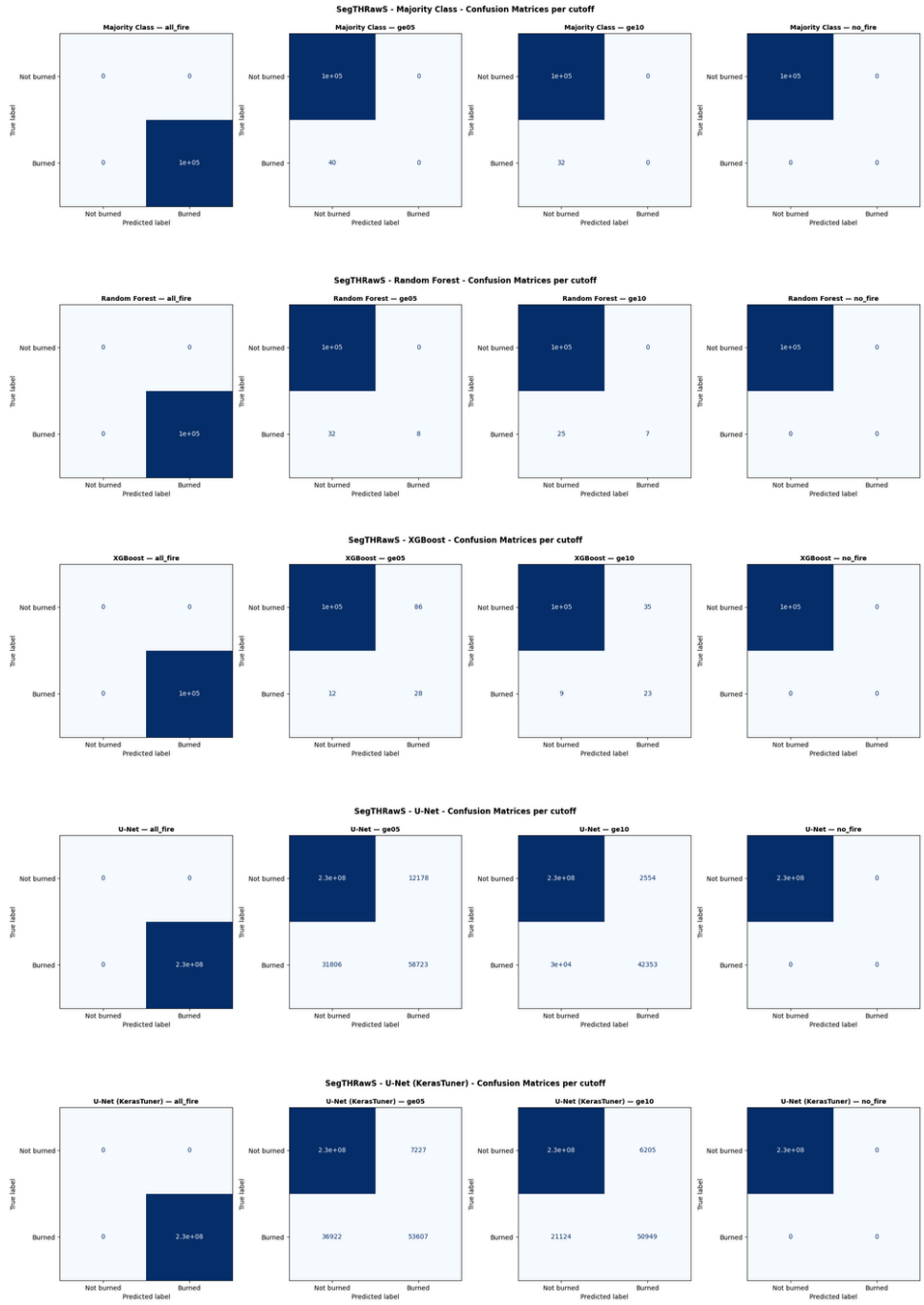

Figure 10: Confusion matrices for all models at threshold 0.5 across all cutoff definitions for the SegTHRawS dataset

B Use of external tools and AI

This thesis has used LLM's for several tasks during this study. The tools that were used are Claude and Gemini. LLM's were used for the following purposes:

- During the writing process: assistance with structure of English sentences and finding suitable synonyms.
- Together with StackOverflow and GeeksForGeeks, LLM's assisted with debugging during the implementation of the models and data pipelines.
- LLM's were also used for help during the training of the models on the ALICE HPC. For help with making slurm files, submitting jobs, understanding partitions and saving output.

All ideas, concepts and the content of this thesis are exclusively developed by the author.