Radboud University

Msc Thesis

Dam Localization using Object Detection

Author:
Stephan Dooper

Supervisor:
Tom Claassen

January 29, 2020

Contents

1 Introduction

For many centuries dams have been constructed to control the natural flow of water streams, mostly to prevent safety against floods, but also to secure a steady water supply for immediate consumption, irrigation for agriculture, and more recently for hydro-power energy. Nonetheless, dams also obstruct the natural flow of water streams, and block connectivity with other parts of the water network, which can have negative consequences for water quality, ecology, and biodiversity [1, 2]. An example is the effect on water life, such as migratory fish that are dependent on free-flowing river networks to reach their destination. Another case is erosion caused further downstream since dams block the sediments that are otherwise carried further downstream [3]. Different factors thus have to be weighed. However, the trade-offs between the advantages and drawbacks of dam placement are not yet fully understood and can be quite complex, especially if multiple dams have to be considered.

Thus, more research is needed. In order to do so, accurate and complete information on dam locations is necessary. There exist several dam databases, but they vary in terms quality, coverage, and consistency in the definition of what a dam actually is. For example, GRanD features 7320 dams which are larger than 0.1 km^3, and $GOOD^2$, which contains 38000 dams and is currently the most comprehensive global dataset[4, 5]. However, this still falls short as it has been estimated that there are over 800,000 dams worldwide [6]. Despite many ongoing efforts, there is still no globally consistent and database for dam locations. Furthermore, geo-spatial locations of dams are obtained through manual geo-referencing, examples include $GOOD^2$, DRIP and AMBER.[1] The problem with manual georeferencing is that it is time consuming, and often results in incomplete data with biased spatial coverage [7].

Deep learning algorithms excel at audiovisual tasks, and can thus contribute to georeferencing dam locations and help towards creating a globally consistent dam database, as well as alleviate the spatial bias and the time consuming nature of manual referencing. Convolutional Neural Networks (CNN) architectures have shown to be particularly useful, and have even been able to achieve superhuman performance on various classification and recognition tasks [8, 9]. One of the problems of classification networks is that they only tell whether an object is present on an image, but not actual location of the object. As a solution, an object detector can be used to make the leap from classification towards actual explicit localization of the dam object. Object detection algorithms have greatly improved in the last decade, with state of the art architectures such as YOLO, RetinaNet, SSD, and various RCNN architectures. Since both the classification and object detection algorithms are supervised, a substantial amount of examples is required. Thus, several databases of dam locations are combined, and patches of spatial data bands are extracted from the Sentinel 2 satellite using Google Earth Engine. Since the ratio of dam to non dam locations is severely lopsided, many non-dam locations are also

[1] DRIP: https://viewer.globio.info/
AMBER: https://amber.international/

generated so that the class imbalance is maintained at test time, and reflects performance more accurately.

1.1 Problem Definition

In this thesis, the main goal is to create an algorithm using neural network architectures that can classify and localize dam locations, given satellite image input. To this end, various neural network architectures are proposed that have performed well in recent years in other audiovisual tasks. Since satellites have a wide variety of multispectral spatial data bands, a variety of feature representations will be compared. Then, an object detector is created and trained, using one of the compared networks and input feature representations to serve as the backbone of the object detection algorithm. The aim is to show that the detector can indeed localize dams, whilst minimizing the amount of false positives, as class imbalance is a major issue with dam localization. Finally, the network should also be able to partially address the spatial bias inherent in existing datasets by being able to predict dam locations in areas where information on dam locations is sparse. For clarity, this thesis will try to answer the following research questions:

- RQ1: Can object detection algorithms be used to classify and localize dam locations using satellite imagery?
- RQ2: Can we increase performance by incorporating different feature input representations?
- RQ4: Can we apply the detection algorithms to (very) large areas, i.e. does the solution scale well to larger input images?
- RQ3: How well does the approach generalize to large areas and locations with low data coverage, i.e. handle spatial bias?

2 Related Work and Approach

As stated in the introduction, the main focus of this thesis is using object detection in combination with remote sensory data from satellites. In the last decade, the area of object detection has improved significantly with the rise of convolutional neural networks becoming being the de facto standard for audio visual recognition tasks, given rise to architectures such as YOLO, RCNN, and SSD [10, 11, 12, 13, 14]. YOLO architectures pose object detection as a regression problem, where the input image is divided into a grid, and where each cell predicts a number of bounding boxes. In contrast, modern RCNN architectures such as Fast(er) R-CNN use a region proposal network, where a convolutional neural network first extracts promising regions, and then uses a combination of a classifier and a regressor for classification and bounding box refinement, respectively. Single Shot Detectors (SSD) add extra feature layers to a backbone network. Then, each of the feature layers can be used as predictors for detections, using convolutional filters. There are several previous studies that used object detection algorithms with aerial or satellite data. For example the authors in [15] used a two stage detection model in a whale counting problem consisting of two CNN architectures. Essentially a large high resolution image is cut into a grid, then each grid cell is evaluated by the first CNN, which acts as a classifier. If the grid cell is estimated to have a whale, it is passed on to a Faster R-CNN architecture, which is an object detector that counts and localizes the whales. This approach mainly serves to make the use of object detectors more feasible in large images, as only a small subset is actually evaluated by the object detector. Another study involving the use of high resolution aerial imagery and object detection is for example the one done in [16], where an adjusted Fast R-CNN network was used in ship detection. This adjustment consisted of changing the ROI pooling layer to account for oriented bounding boxes, where the ground truth boxes are not longer strictly horizontal or vertical. There are also several applications involving YOLO architectures, such as [17, 18, 19] where YOLO v2 and YOLO v3 architectures are used on ship and vehicle detection problems. A drawback of these studies, is that they do not make use of multispectral channels that many satellites offer, which is also mentioned in [20]. Such data bands can be valuable to use as they can enrich the feature representation and help improve classification performance. This is illustrated in the work of [21], in which the authors proposed using multispectral Landsat-8 images for solar power plant detection, and showed that adding extra signals can provide significantly different values between target objects and background, whereas standard color channels would fail to do so and add no extra discriminatory ability.

2.1 Top Level Approach

As already mentioned in the introduction, the main objective of this thesis is to create an object detection framework that is able to detect dam locations using remote sensory images. To accomplish this, the Google Earth Engine API is

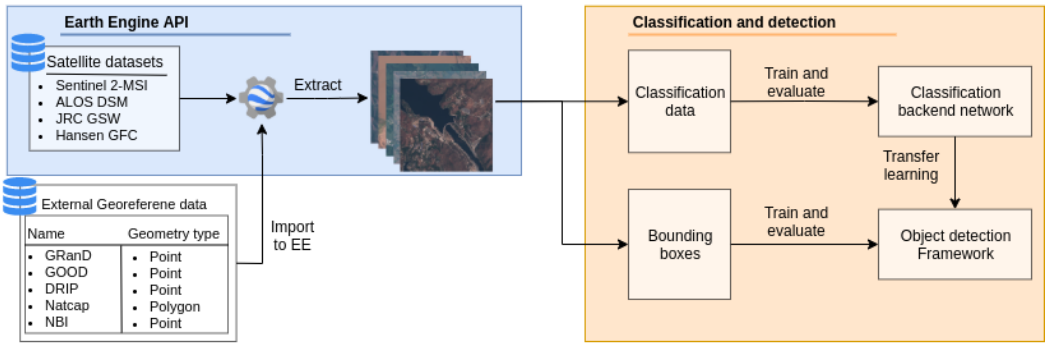

Figure 1: Top level overview of the data flow. External datasets with latitude and longitude coordinates of dam and bridge locations, along with one dataset with polygon coordinates (natcap), that are bounding boxes around dams. These datasets are imported into the Earth Engine API and then combined with satellite data to extract input features that are fed into the neural network architectures.

used to combine datasets containing geospatial coordinates of dam locations, which are then used to extract image patches that contain multispectral features. Additionally, a dataset containing polygon geometries is also used that contain bounding boxes around dam locations, which can be used for object detection.

The extracted data is then used in two separate stages. First, a classification backend network is trained on a classification task, without any bounding box data. Then, a YOLO v2 network is trained using bounding box data, using the pre-trained backend network from the first stage. The top-level workflow is depicted in figure 1, and is discussed in more detail in section 3. This workflow is similar to the way in which the original YOLO networks are trained [11]. To avoid possible confusion later on, a small summary on the data sources and their origin is given.

2.2 Data Sources

The datasets can be split into two categories: external datasets with either point geometries or polygon geometries, containing lat/lon coordinates for dam and bridge locations. The second category is composed of satellite imagery which is accessible via the Google Earth Engine API (GEE). The former set is described first, then the latter. The first dataset is the Global Reservoir and Dam Database (GRanD), version 1.3 (7320 samples) [4], which maps dam locations greater than 15m in height, or with a reservoir of more than 0.1km^3. The second dataset is the GlObal geOreferenced Database of Dams (GOOD^2)[2] (32613 samples), which has both large as well as medium sized dams. The third dataset is the Dam and Reservoir Inventory Project (DRIP) (3058 samples).[3] Additionally, a dataset of dam locations along with bounding boxes is used, which consisting of 45214 samples, which is dubbed Natcap. Finally, bridge

[2]The global dam watch claims that GOOD^2 has over 38000 dam locations, but the Good2 unsnapped database file only has 32613 records when downloaded.

[3]`https://viewer.globio.info/index.php`, accessed 5-11-2019.

locations are extracted from the National Bridge Inventory (NBI) [22], using only samples with a navigational control code value of 1 [23]. This leaves us with a dataset of 4136 bridge samples.

The second source of data is accessible via the GEE API and consist of satellite images. In this thesis, we mainly use data from the Sentinel 2 (S2) satellite (Level-1C orthorectified top-of-atmosphere reflectance) [24]. Additionally data is also gathered from the ALOS DSM, JRC Global Surface Water (JRC GSW), and Hansen Global Forest Change (HGFC) satellite images [25, 26, 27, 28].

The reason why these datasets are used is twofold. First, several of these datasets are used for sample *extraction and generation*, e.g. to extract dam locations, or to generate new locations that serve as negative sample (non-dam locations). The datasets that are used for this purpose are GRanD, GOOD2, DRIP, the bounding box data, NBI, JRC GSW, and HGFC. The second purpose is to *extract features* that serve as input to the neural network architectures. The S2 satellite and ALOS DSM dataset are used for this specific purpose. An overview of the satellite dataset can be found in table 1.

Sensor	Band	Wavelength S2A/S2B (nm)	Res (m)
S2	B2	496.6 / 492.1	10
(2017-01-01 / 2019-11-15)	B3	560 / 559	10
	B4	664.5 / 665	10
	B11	1613.7 / 1610.4	20
ALOS DSM	AVE	-	30
HGFC	treecover2000	-	30
JRC GSW	occurrence	-	30

Table 1: Satellite image data overview

3 Methods

The main methodology to tackle dam detection involves using an object detection algorithm (YOLO V2), which is trained in two steps. The first step involves choosing and finetuning a backend CNN. The second step uses this backend to extract features, and passes them to a detector network, which also has to be trained.

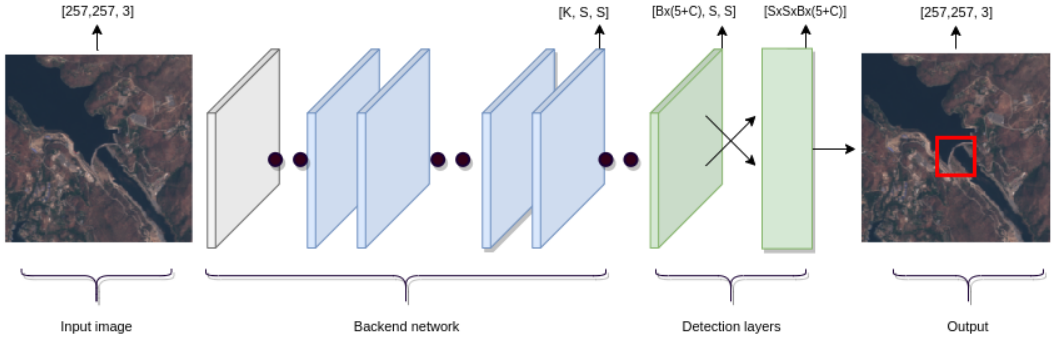

Figure 2: Object detector framework. The input image is passed through a replaceable backend network and serves as a feature extractor. The final output of the backend network has K feature maps, and a resolution of SxS. The output from the backend network is then passed through the object detection layer, which is a convolutional layer with 1x1 convolutions and has $B \times (5 + C)$ feature maps and a resolution of SxS. This output is then reshaped into a 3D tensor.

3.1 YOLO v2

The YOLO v2 architecture was proposed by Redmon et al. in 2017 and poses the object detection as a regression problem [11]. YOLO divides the image into a grid of size $S \times S$ and predicts B bounding boxes with C class probabilities $\mathbb{P}[class_i|\text{object}]$ for every grid cell. Each bounding box predicts the center of the object t_x, t_y and the width and height t_w, t_h, both relative to the location of the grid cell, and a confidence score t_0, which is formally defined as $\mathbb{P}[\text{object}] \times \text{IOU}^{truth}_{pred}$. Finally, offset predictions are created via the following formula. Let (c_x, c_y) denote the offset from the top left corner of the image, and (p_w, p_h) denote the width and height of the anchor box $b \in B$, respectively, then the predictions yield

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \\ \mathbb{P}(\text{object}) \cdot IOU(b, \text{object}) &= \sigma(t_0) \end{aligned}$$

where σ is the logistic function. Altogether, each grid cell makes a total of $B \times (4 + 1 + C)$ predictions, where 4 equals the box information, and 1 equals the confidence score for the box. A high level overview of the object detector

is given in figure 2. during training, the YOLO v2 network optimizes the MSE loss between the predicted bounding box and the ground truth box over several factors. The loss is defined as

$$
\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} 1_{i}^{obj} \left(p_i - \hat{p}_i \right)^2
\end{aligned}
$$

Predictions from the YOLO network are finally post processed according to two steps. First, bounding boxes with low confidence predictions are filtered out using a threshold score. Then, a non-maximal suppression algorithm is applied to remove any overlapping bounding boxes that point to the same bounding boxes in the following way:

1. pick a threshold
2. Remove all boxes with a probability lower than the threshold
3. while there are still boxes, pick the box with the largest probability, and discard other boxes that have an IOU greater than with the previous box

3.2 Backend Architectures

The YOLO V2 network allows for some degree of freedom when it comes to choosing the neural network that is used for extracting high level features. This backend network can drastically alter the performance of the entire object detection framework, which is usually measured in terms of the mean Average Precision (mAP) and the efficiency (the speed at which it can digest the images), usually measured in frames per second (fps). Since the latter measure is less important for the application in this thesis, computationally heavier backend models can be utilized. In this thesis, several different backends are evaluated.

First, the original YOLO backend network is used, which is called Darknet19. This is a fully convolutional network that repeats several convolutional and max pooling operations. Additionally, we also add three repeated 3x3 convolutional layers with 1024 filters each, followed by a final 1x1 convolutional

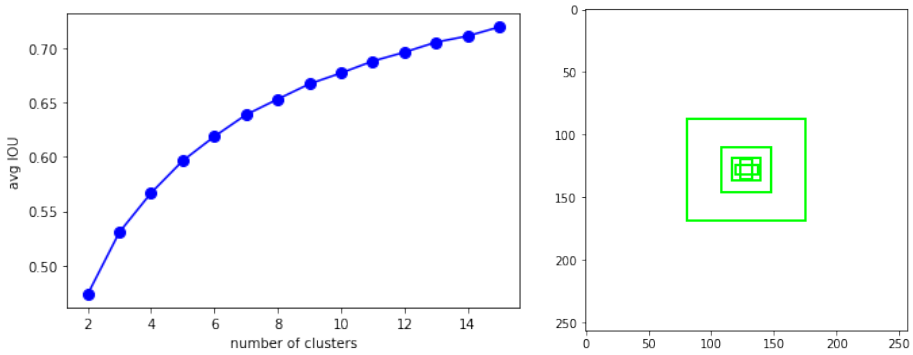

Figure 3: K means algorithm performed on ground truth box dimensions. The left plot shows the average IOU compared to the number of centroids K. The right plot shows the obtained anchor boxes for $K = 5$.

layer, with the number of outputs needed for detection. This is also done in the original YOLO v2 approach. This network serves as a baseline, and is compared to several other models.

The second model that is used is a ResNet-50 architecture. This network consists of residual blocks, where each block has 3 layers. These three layers consists of 1×1, 3×3, and 1×1 convolutions, with varying number of filters, depending on the depth of the residual block. For a more detailed description of the network the reader is referred to [29].

The final architecture is a DenseNet-121 network [30]. A DenseNet is composed of several dense blocks, which in itself consists of several layers. Within each dense block, each layer takes all preceding feature-maps as input via a concatenation operation, rather than summation. Between each dense blocks, transition layers are included to allow for downsampling, which consist of a batch normalization layer and an 1×1 convolutional layer followed by a 2×2 average pooling layer.

Finetuning

To facilitate the training procedure for the backend networks defined previously, pre-trained ImageNet weights are used. Since several combinations regarding finetuning can be made, three different training strategies were tested, namely training from scratch without any pre-training, using pre-trained weights and freezing all but the top layer, and using pre-trained weights but not freezing any of the layers. Of these tests, using pre-trained weights without freezing achieved the best results for classification.

3.3 Tuning Anchor Boxes

The number of anchor boxes have to be chosen manually. The authors in [11] noted that clustering algorithms give much better results than using hand-picked priors. As such, we follow their clustering approach, which involves using a k-means clustering algorithm on the training set with the following

distance metric

$$d(\text{box, centroid}) = 1 - IOU(\text{box, centroid}) \tag{1}$$

The k-means algorithm is run for various values of K, with 10 retries to allow the algorithm to find better local optima. The average IOU to the number of clusters is shown in figure 3, and the average IOU with the closest centroid is plotted. A tradeoff now has to be made between model complexity and high recall. Throughout this thesis, a value of $K = 7$ will be used.

4 Data

In order to train neural networks, large amounts of data are needed with appropriate discriminative features. The datasets are extracted using Google Earth Engine (GEE), which provides several petabytes worth of satellite data, and an environment for data analysis. The GEE environment is used to extract spatial data bands from several different satellites, and to sample from several locations across the planet in order to obtain a rich collection of diverse locations.

Finding dam locations is not a trivial problem: The earth roughly has a surface area of 510.000.000 km^2, and even if we discount the areas covered by the oceans where surely no dams reside (71%), then there is still around 147900000 km^2 of the earth's surface where dams can be located. Of course it is possible to lower this number even more by constraining ourselves to lakes, rivers, and other water bodies inside land masses, but the idea should be clear. As such, there is an inherent class imbalance. Another problem is the planet's diversity: the earth has a wide variety of sceneries that make up different landscapes, ranging from dense forests, high mountains, vast plains and deserts, etc. This means that there is no one size fits all solution for sampling: we cannot expect to sample for a single small region, country, or even continent, and expect it to generalize well. What is needed is a dataset that covers places from all over the globe, both with dam locations and non dam locations.

Besides dam locations, examples without dams are also necessary, and several categories are created. These categories are forests, locations near edges of water bodies, bridges, and miscellaneous locations. These categories were included either through preliminary testing, in which was observed that the backend model classifier gave high probability to places where it should not, suggesting that more examples were needed, or because we thought beforehand that the classifier might have a hard time dealing with such areas. A top level overview of the data preprocessing steps is given in figure 4.

4.1 Preprocessing and Sampling

Dam and bridge locations can be sampled directly from the GRanD, $GOOD^2$, DRIP, NBI, and bounding box datasets, without any further preprocessing. Other non-dam locations are generated First, forest regions are extracted by using the treecover band of the HGFC dataset. In order to filter for feasible regions with a lot of tree coverage. A threshold parameter is used, and only regions that contain 80% or more are kept for sampling.

Secondly, edges of water bodies form a source of interest, since they might be hard to classify. To this end the JRC GSW water occurrence band (%) is used, which contains water occurrence of each continent on the globe, excluding oceans. A Canny edge detector is then applied on the water occurrence band with a threshold parameter of 60 [31]. This method works well since water occurrence tends to drop sharply at the boundary between a water body

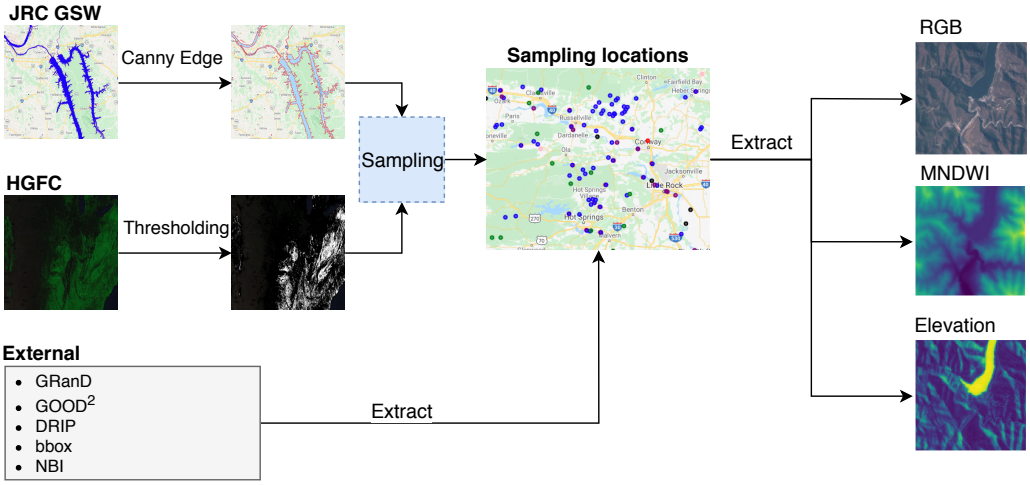

Figure 4: Flowchart of dataset preprocessing operations.

and dry land, and can thus be detected as edges. The threshold parameter determines whether a gradient value (here, a change in water occurrence) is detected as an edge or not. Higher threshold values result in gradient values to be classified as edges. The sigma parameter is kept small so that fine scale features are also detected (edges of small water bodies tend to be underrepresented with high σ). Water boundary samples are then collected by randomly sampling over the obtained edges.

Finally, a large amount of samples are randomly drawn from the earths surface in order to account for a multitude of other categories not specified above. This way, other regions are still represented in the dataset that are not accounted for manually.

4.2 Feature extraction

Satellite imagery offer the advantage that multispectral channels and other features are available. In this thesis the standard RGB features are used, along with elevation and water index features. The RGB and water index features are extracted using the S2 satellite data. A common preprocessing step with S2 satellite images is to filter out obstructions in the images, i.e. clouds. The way this is done is by first selecting images within a specific date range (see table 1). Then, granules are selected that have less than 20% cloud cover, over which the median is calculated over time. By doing this both cloud obstructions are minimized, and heavy color variations that might appear when two images are taken at different times are smoothed out.

Water index feature can help discriminative water from surface by using a combination of colour channels, as well as infrared channels. This can be useful, as dams are located in or near water. The most common water index is the Normalized Difference Water Index (NDWI) [32], however, a modification was proposed in [33], where instead the ShortWave InfraRed (SWIR1) is used

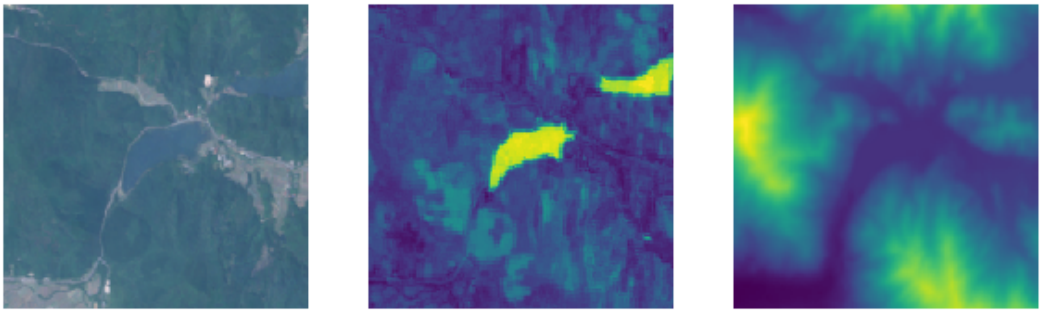

Figure 5: Features from left to right: the RGB image, MNDWI heatmap, elevation heatmap.

instead.

$$MNDWI = \frac{Green - SWIR1}{Green + SWIR1}. \tag{2}$$

This feature obtained higher validation scores over the regular NDWI, and will thus be used in this thesis. The final feature that is used is elevation. This feature is obtained by using the ALOS Digital Surface Model (ALOS DSM), which is sampled at 30 meter resolution. To match the native resolution of the other spatial data bands, the elevation feature is downscaled to 10 meter resolution using bilinear interpolation. An example of the extracted features is shown in figure 5.

4.3 Patch extraction

Using GEE, patches with a size of 257 $\times$ 257 are extracted using the data sources and sampling and feature extraction procedures defined in sections 4.1 and 4.2. The patches are extracted around the center of the latitude and longitude coordinates for each sample. Samples were generated all over the globe whenever possible, so that spatial bias is minimized and a diverse set of examples is created. The final classification dataset is imbalanced with a class ratio of 0.1, 0.9 in favor of non dam locations, and is split into a training, validation, and test set according to table 2. The distribution was chosen in such a way that the training set remained fairly balanced. The validation and test sets, however, both exhibit class imbalance to mirror the real world situation wherein dams are extremely rare ($< 1\%$).

The reason for this specific split is that a simplification step was made: first, we wanted to show that dam classification is indeed possible under a mild form of training imbalance, and a heavy form of validation and testing imbalance. Secondly, the main goal of the classification scheme is to learn high level features, so that it can be used in the YOLO v2 network.

For the bounding box dataset 20% is set aside to use as the test set. Of the remainder, another 20% is used for validation, and the rest is used for training.

	Dataset split			
Features	Train	Valid	Test	Total
Drip	2542	100	300	2942
GRanD	6563	100	500	7163
GOOD^2	29240	200	3000	32440
Bridges	1513	200	2300	4013
Forests	5956	300	7133	13389
Water edges	3266	200	4235	7701
Random	37720	2400	55774	95894

Table 2: Data set distribution and splits.

4.4 Data augmentations

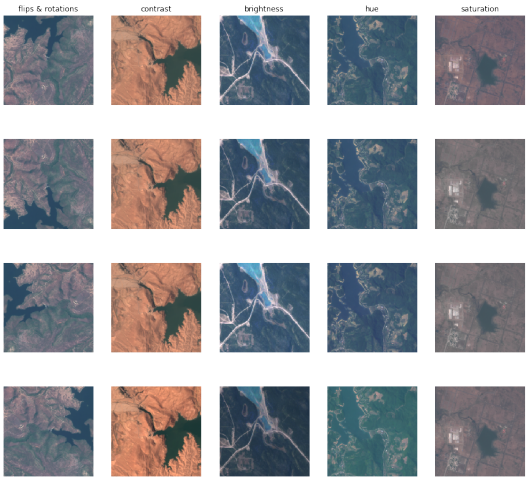

Figure 6: Data augmentations from left to right: flips and rotations, contrast, brightness, hue, and saturation.

Data augmentation is used to incorporate further diversification at training time. The set of transformations used consisted of flips (left, right, up, and down) and rotations (set of 90, 180, 270 degrees rotations), scaling, zooming, shears, translations, contrast and brightness perturbations, as well as changes in colour space by altering hue and saturation. The values for the augmentation parameters for the latter 4 transformations were chosen through manual inspection, and further refined by optimizing the F1 validation score. The reason for also opting for colour augmentations is due to the fact that satellite images can be taken at different times, and depending on a variety of factors (time, weather, etc), the colour and brightness of the same location can vary.

5 Experiments

As explained before, the main procedure involves creating an object detector using two neural networks, and training them in a two stage fashion. In the first stage, a backend network explained is trained as a classifier using pre-trained ImageNet weights. This can help to improve results when training the detector in the second stage, as the backend network will already have learned to identify dams, and is thus able to high quality features. In the second stage, the object detector is trained by removing the prediction layers of the backend network and chaining the remainder to the detector layers, see figure 2. It is important to note that none of the layers are frozen during this stage. This is because the object detector is trained using ground truth boxes, rather than classification labels. The backend network should thus still have the freedom to adapt to the new learning objective.

5.1 Backend Networks

The chosen backend networks are defined in section 3.2. These networks are first fine tuned to a dam classification task with the dataset defined in 2. Since there are several features available, and it is not obvious which representation would yield the best results, an ablation study is performed with the standard RGB channels as a baseline. Each model is instantiated with its respective ImageNet weights and then trained for 10 epochs using an ADAM optimizer with learning rate $1e-4$, and a batch size of 32. Furthermore, to alleviate the small class imbalance that is present in the training set, random undersampling and oversampling is used on the majority and minority class, respectively. Finally, the augmentations are performed on the training set in the procedure described in section 4.4. All of the models are trained on the training set, and their performance is validated and tuned on the validation set. The final performance is tested on the test set.

5.2 Object Detector

Object detectors have proven to be hard to train. This is partially due to the fact that YOLO v2 is posed as a regression framework. This particular design means that the outputs of the network are unbounded. Divergence was a common occurrence during training, where the coordinate loss tended to explode in several runs. The solution to this was to run the network for several epochs in a warm up period fashion with a very low learning rate so that the network weights were allowed to stabilize. During this period, the learning rate would gradually increase until it reached the end of the warm up period. From that point onward, an exponential decay learning rate is used to further fine tune the networks. An example of this learning rate scheduler is shown in figure 7, which is the same for every backend network used throughout the thesis. All models were trained for 50 epochs, with a batch size of 10.

Additional parameters that need to be tuned are the scales for the loss function and. After several runs, the best validation results were obtained

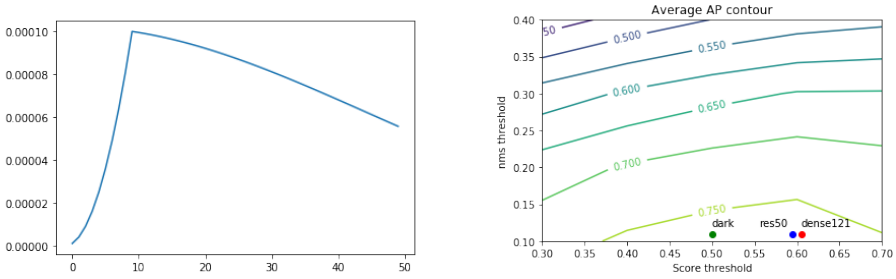

Figure 7: learning rate schedule for the object detector, and average contour for the AP results over a score and nms threshold grid with the optimal values for each backend network

by using $\lambda_{obj} = 5, \lambda_{noobj} = 1, \lambda_{coord} = 1$. The number of anchors for each backend network was chosen to be $K = 7$. Finally, the threshold parameters for the confidence scores and NMS algorithm need to be tuned as well. For each trained backend network, several combinations of the confidence scores and iou threshold were tried and evaluated against the AP metric. For all networks, an nms threshold of 0.1 provided the best results. For the DenseNet-121 and ResNet-50, a score threshold of 0.6 is obtained, and for the Darknet19 a score threshold of 0.5 is used. After all the parameters were fine tuned, each model was tested on the test with the model that achieved the best validation AP score.

5.3 Hardware, Software, and Monitoring

All models were trained on a single machine with a a single NVIDIA Tesla M40 GPU. All of the experiments were implemented using Tensorflow (v 2.0.0) and Python 3.6. The metrics from the models, such as the epoch losses, were monitored using Tensorboard and Omniboard.

6 Results

6.1 Evaluation Metrics

For the backend network, the F1 metric is used since the dataset is imbalanced, and measures such as accuracy will not correctly reflect the performance of the models. The F1 measure is defined as

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{3}$$

For the object detector itself, the Average Precision (AP) is used as an additional measure. The average precision is calculated as the area of the precision-recall curve

6.2 Backend Networks

The F1 scores for the backend model with the different feature combinations are shown in table 3. The highest F1 score is obtained by using a ResNet-50 model including all of the features. As a general observation, it appears that the F1 score improves when more features are included beyond using RGB channels. This indicates that the models indeed benefit from using additional spectral features such as SWIR1 and elevation. For example, the feature set $RGBW$, and $RGBAW$ both perform better than the baseline RGB features $(0.8259, 0.8108, 0.7979)$ vs $0.8457, 0.8697, 0.8779)$.

	Model		
Features	Darknet19	DenseNet-121	ResNet-50
RGB	0.8259	0.8108	0.7979
RGBA	0.8211	0.8459	0.7701
RGBW	0.8452	0.8500	0.8423
RGBAW	0.8457	0.8697	**0.8779**

Table 3: F1 scores for Darknet19, ResNet-50, and DenseNet-121 networks, using different feature combinations. These scores are based on the test set defined in table 2.

The only discrepancy in this trend is that the $RGBA$ features perform worse than the RGB features for the Darknet-19 and ResNet-50 models, which provides some evidence that it might act more as a noise feature. However, the $RGBAW$ feature set performs better than the $RGBW$ for our performance score, which seems peculiar. For example, the ResNet-50 drops by 0.0278 when adding elevation features to the baseline. However, the same did not hold when adding them to the $RGBW$ features, and instead noted an improvement of 0.0356. A similar pattern emerged when the Darknet-19 scores are observed. A possible for reason for this behaviour could be that the elevation features themselves might not always be an improvement, but when combined with

both *RGB* and MNDWI features, can serve as a complementary variable that improves the overall classification score. [4]

Figure 8 shows the class activation maps using Grad-CAM for the backend models [34]. Most of the activations seem to be clustered around the center of the image in a star or 'plus' shaped pattern. One of the few exceptions are the examples in the final column, which have relatively high activations throughout the entire image especially for the ResNet-50, indicating that the network has problems finding the dam location. One of the possible reasons for this could be that it is an image with a large tree coverage, and that the network still has significant difficulties in forest areas, despite the included forest training examples. We would like to argue that it is unlikely that the model mistakes forest regions for water, due to the inclusion of MNDWI features. The DenseNet-121 (middle row) shows similar results, but has square shaped activation values, which are also spread across a larger area, which might mean it is less precise in accurately pinpointing the exact dam locations, but rather points a general area of interest. Finally, the Darknet19 (bottom row) has less defined shapes for its activation values, which might stem from its lower accuracy results and that it has not learned to pinpoint dam positions as precisely as the former two models.

[4] It could also be that the model needed more training epochs to fully converge. However, the validation metrics declined sharply after 11 epochs, making us believe that overfitting had set in.

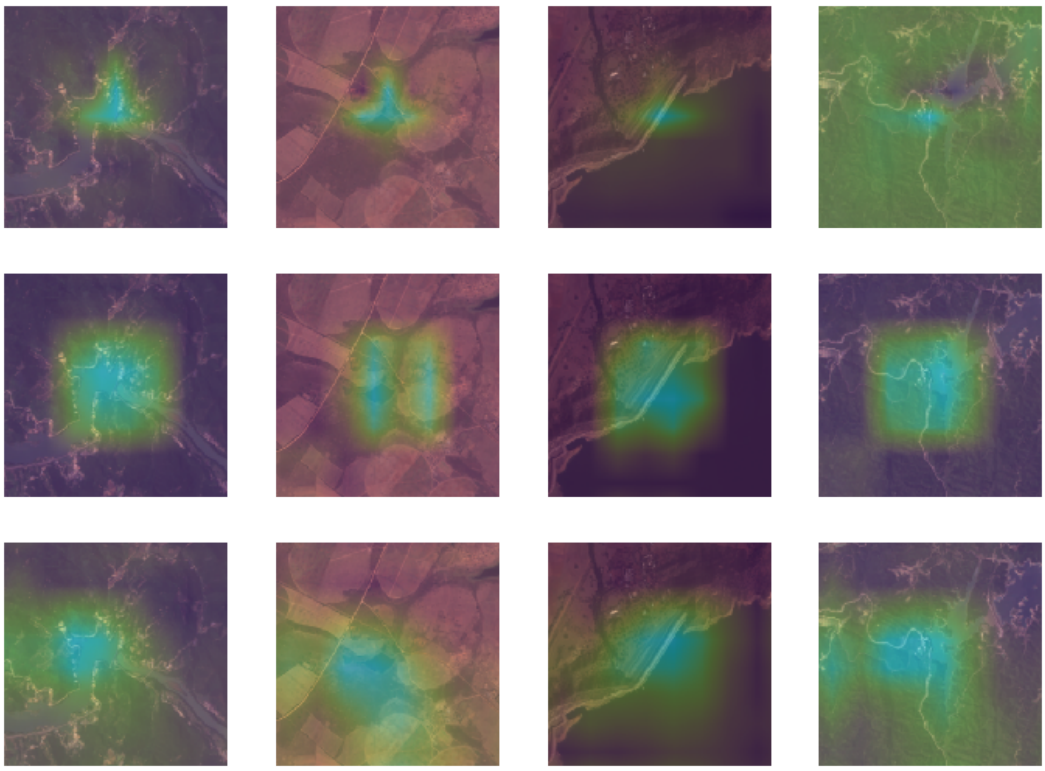

Figure 8: Grad-CAM for ResNet-50 (top), DenseNet-121 (middle), and Darknet19 (bottom). All of the results are derived from the models using all of the features (RGBAW

6.3 Object Detector

The results for each backend network is shown in table 4, where the AP, F1, precision, and recall scores are shown for each backend. In terms of AP, the DenseNet-121 backend obtained the highest score (0.7891), being slightly higher than the ResNet-50 (0.7889), and higher than Darknet19 (0.7540). The ResNet-50 performs best in terms of F1 and precision scores, although it has the lowest recall score out of all the three models. The observation that the ResNet-50 has a much higher precision score (0.9238) than the other two models (0.85596, 0.8538) is more or less in line with the earlier observation from figure 8, which showed that the class activation maps for the ResNet-50 were more focused around dam areas than the other networks. Finally, the DenseNet-121 has the highest recall score. In short, in terms of AP, both the DenseNet-121 and ResNet-50 seem to be improvements over the Darknet19 backend, which is not entirely surprising given the classification results from table 3, and since it was primarily designed for real time object detection, i.e. high speed.
Several detection results from the test set using the DenseNet-121 backend are shown in figure 9. Contrary to the usual case where the number of ground truth boxes coincide with the number of objects in the image, the images in the current dataset can have more dams than bounding boxes. This is due to the fact that annotating ground truth boxes is a hard, exhaustive, and time

	Model			
Model	AP	F1	Precision	Recall
Darknet19	0.7540	0.8500	0.8538	0.8463
DenseNet-121	**0.7891**	0.8650	0.8596	**0.8704**
ResNet-50	0.7889	**0.8763**	**0.9238**	0.8335

Table 4: Object detection AP results

consuming task. For this reason, on the top left of each image the ratio of predicted boxes to ground truth boxes is shown. The ground truth boxes are shown in colored filled rectangles, and the predicted bounding boxes are shown as green rectangles. In most cases, the detector is able to predict the location of the dam. We note that the detector does seem to have some difficulty properly fitting the exact location and shape of the bounding box itself in several cases (an example being row 2, column 1). However, an argument can be made that this is not necessarily a bad thing, since the aim is to predict the location of the dam, and we are not generally interested in the actual shape of the dam or the bounding box. A second observation worthy of noting is that the algorithm is able to detect smaller objects as well, which YOLO usually has difficulties with [10, 11]. This could still happen when multiple small objects are within the same grid, since then YOLO would not be able to detect them, since they are assigned to the same anchor box. Finally, an example where it is shown that the ground truth dataset does not necessarily cover all dam locations within each patch, can be found in the first row and second column of figure 9, where at the bottom left corner, there turned out to be another dam location, but the ground truth did not cover it.

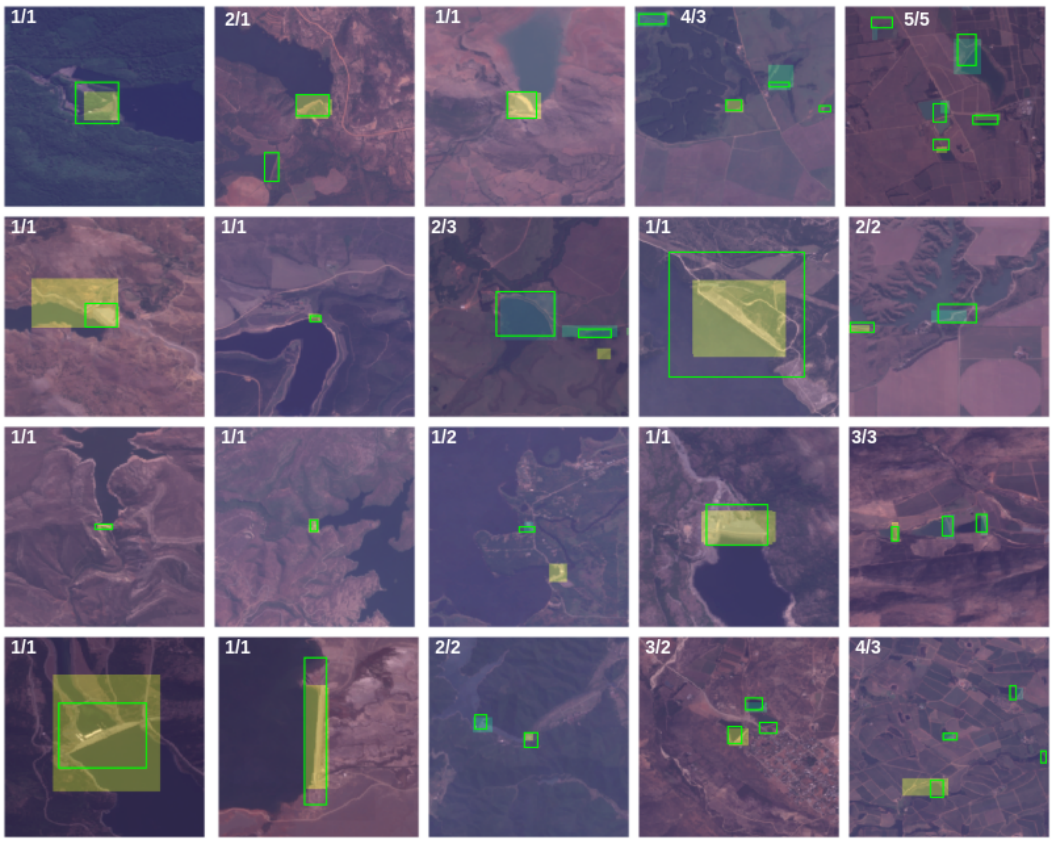

Figure 9: Object detection results from the test set. The predicted bounding boxes are depicted in green. The colored segmentations are the ground truth bounding boxes. The white numbers represent the number of predictedboxes /number of ground truth boxes.

6.4 Detection in large areas

The results in sections 6.2 and 6.3 have shown that the approach does reasonably well on simple patch classification and object detection tasks. However, these are only solutions to smaller problems of the actual animal that needs to be tackled: detecting dams in large areas.

Analyzing large areas can potentially span many kilometers, and at the same resolution of 10m, the input images size will far exceed the patch size of 257×257 that was used to train all of the previous models. Thus, there are several hurdles to overcome when switching from patch classification and detection towards analyzing larger areas. First, the models were trained on small patches, and it is not trivial to say that they will do well for larger input sizes. Resizing the large image into a (much) smaller one is also not a good idea. This counts especially for YOLO architectures, where the image is divided into a grid, and where each grid cell has a maximum number of objects it can locate. If a large area with potentially many dams is resized into a smaller image, then YOLO will not be able to locate them due to the sheer amount of dams in each grid cell, which are all resized to an incredible small size. Finally,

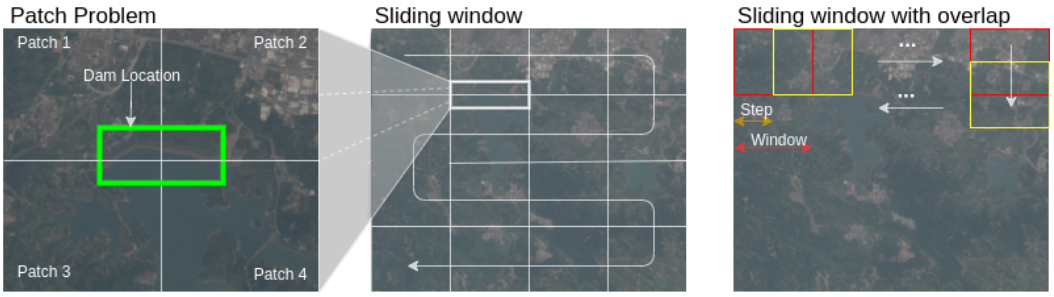

Figure 10: from left to right: the patch problem, the original sliding window procedure described in [20], and the proposed adjustment with overlapping windows. In the patch problems, the original dam is cut into several pieces, and becomes unrecognizable in every individual patch.

there is also the problem with computational feasibility: large input images require large amount of memory, and the processing time will also be longer.

In order to tackle these problems, and make full use of all the models that have been trained and tested up until now, an adjustment to the method described in [20] is used. In short, a large image is analyzed in patches using a sliding window with a step size equal to the patch size, where each patch undergoes a classification stage, and optionally a counting and localization stage. In the first step, the patch is classified using the best classifier from section 6.2. If the dam probability from the classifier in the patch is greater than 0.5, the patch will be analyzed by the best object detector from section 6.3, in which the dams will be counted and localized. For the remainder of this section, the classification will be handled by ResNet-50 classification network, which had the best F1 score as discussed in section 6.2, and the YOLO v2 with the DenseNet-121 backend network, which achieved the highest AP result.

The method described above worked quite well to detect whales, for which it was originally designed. However, the only problem is that the target objects in the original paper are much smaller than the image patch size. That way, the probability of an object being divided over multiple patches and therefore becoming unrecognizable was slim. However, this is not the case for dams, and they are more often being severed into several patches so that the dam itself is missed. Therefore, we slightly adjust the procedure by allowing for overlapping windows. We use patch sizes of 257×257 along with a step size of 128. The difference in procedures is outlined in figure 10. This method has several advantages over e.g. a single step approach where only a single stage object detector is used to classify and detect dam locations using a sliding window. First, there is an advantage in terms of speed, since object detectors are usually slower than classification networks. Secondly, the use of the classification network allows for some degree of selection, filtering out and separating potentially interesting areas, which might then be further analyzed by the object detector. This could drastically reduce the amount of false positives

that would otherwise be generated by the object detector.

The adjusted procedure is tested on different regions in Germany, Zimbabwe, and Brasil. These regions have several different characteristics, and the algorithm also performs differently in each case. Figure 11 displays the detection results for the Raffingora region in Zimbabwe, which is for the most part a rural area. This region has many small dams, totalling at 32 dam locations. The water is quite easily discriminated from dry land from the RGB bands alone since it is much darker than the dry land around it. Out of all the labels, the algorithm is able to predict 17 (or 18) correctly and accurately localizes them, out of a total of 28 predictions. Out of the remaining false positives, the algorithm at least manages to pinpoint several promising locations in or near water bodies, without pinpointing every single water body as a potential dam area. For example, the river stream in the bottom left contains water, but no dam.

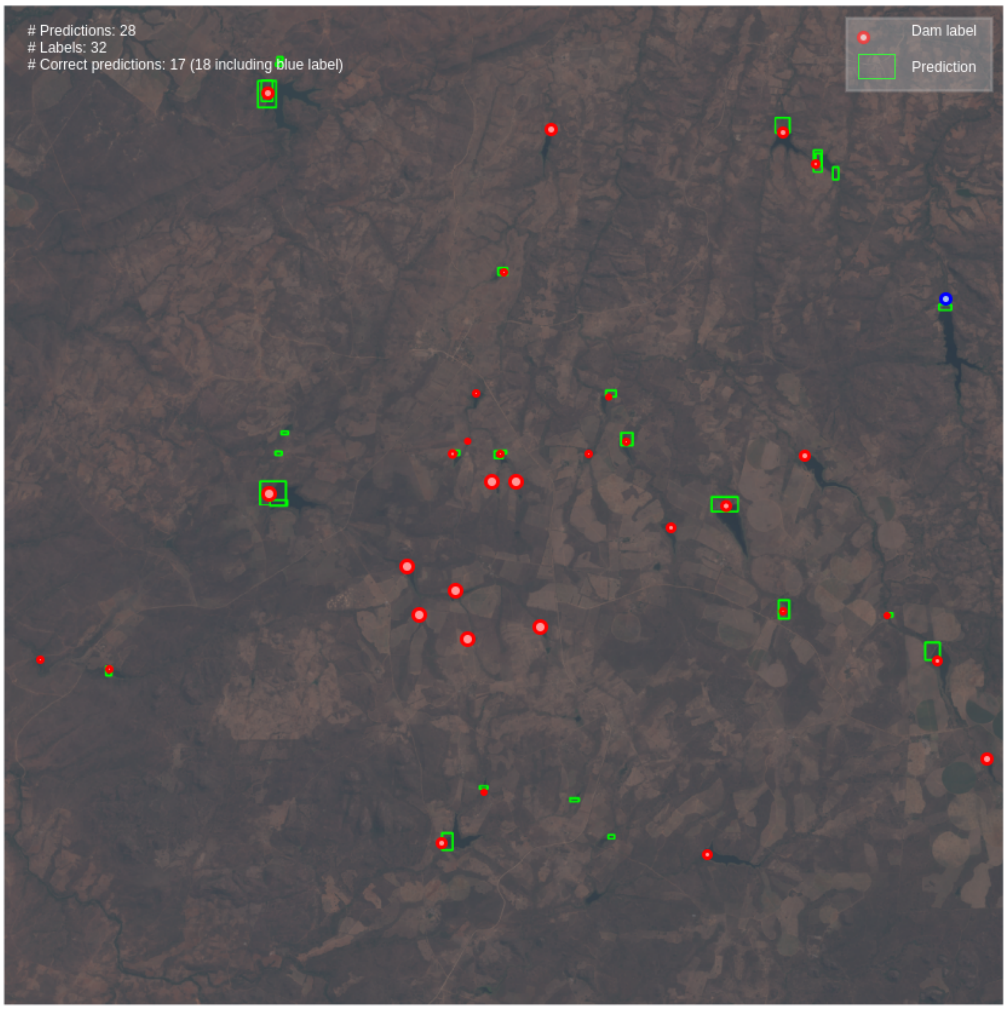

Figure 11: Dam detection results for the Raffingora region in Zimbabwe, which is an area of 23.13×23.13 km (2313×2313 pixels). The red dots are actual dam locations, the blue dot is a dam location that is classified by the author.

The second test region in Germany is different from the first area in several ways. First, it features far more vegetation in the form of trees, which makes it harder to discriminate water from dry land using RGB bands. Secondly, there are only 5 actual dam locations. The detection results are shown in figure 12. This time, there are a total of 6 detections, of which 4 are correctly predicted. This means that the algorithm missed only one dam location, but made 2 false positives, of which one is located in the same water body as an actual dam location. In overall, we argue that the algorithm did its job fairly well in this case, since the area contains many pitfalls such as dark areas which can be confused as water bodies.

Figure 12: Detection results for the Goslar region in Germany, which is an area of 38.5×23.13 km (3855×2313 pixels).

The final test region is located in Brasil, and this region primarily poses as an example where the proposed methodology and algorithm can still be improved to a large extend. The detection results are shown in figure 13. Although the one and only dam location is successfully localized, it also generated a large amounts of false positives. In our eyes, this is most prominently evidence that the classifier fails to accurately discriminate promising locations from background, as detections only occur when the dam likelihood is larger 0.5. In many cases, detections are made far away from actual water bodies. Although there is no conclusive answer as to why this happens, since the MNDWI features should largely mitigate such errors, it could be that the classifier still mistakes some dry land regions for water, or that specific shapes on the land are reminiscent of dam characteristics.

Figure 13: Detection results for the Paraibuna region in Brasil, spanning an area of 25.7×23.13km (2570×2313 pixels).

A final observation that is noteworthy is that the step size parameter can influence the result of the number of detections and quality of the detections to a certain extend. In general, larger step sizes yield less detection results, whereas smaller step sizes give more detection results. However, the drawback is that it also will increase the probability of false positives, as well as detection redundancy, which means that the same object is located twice since the window has not moved away far enough from a promising dam region. This seems to be the main limitation of this approach, meaning that the step size can heavily influence the detection results. The only possible solution to this problem is to recreate the conditions of [6], which means that the target object has to be much smaller relative to the patch size.

7 Discussion

There are still several

7.1 Data quality and split

GEE exports image patches from the center of the specified location. This means that in many, if not all of the patches, the dam was located in the middle of the image. Although this makes for easier training, it is not realistic to assume that the object of interest will always be centered in the middle. Although heavy data augmentation involving zooming, cropping, and sheering can alleviate the problem, it would have been better if off-centered dam locations was also available. Secondly, the results from section 6.2 show that CNN architectures are able to perform dam classification. However, the way the data split was set up made it much easier to train the classification architectures since only the validation and test sets are imbalanced. The next step in this process should be to incorporate a fully imbalanced training scheme that fully reflects the imbalance problem.

7.2 Model performance

Several experiments were conducted to test each individual part of the full procedure. The first part consisted of training a number of backends on a classification task. We have seen that the backends are able to classify dam image patches reasonably well with the ResNet-50 achieving the best F1-score, and that the networks definitely benefit from adding features, such as water index and elevation data, as shown in table 3. The backend networks were then used using transfer learning to an object detection task, wherein the second step was made: counting and localizing dams using a YOLO v2 object detection network. The DenseNet-121 achieved the best AP performance (0.7891), with the ResNet-50 being a close second, and outperforming the former model in terms of precision.

Finally, the classification and object detection architectures were combined in order to classify larger areas. Although several of the results give fairly good results, namely for the Goslar region in Germany, and to a lesser extend the Raffingora area in Zimbabwe, we also concluded that there is definitely room for improvement, since the results were still poor for the Paraibuna region in Brasil. A major factor in this is that the classification network does not act well enough as a filter, as it classified too many areas within the regions containing dams, which resulted in a high number of false positives. Possible ways to improve on this is to include even more negative examples, or to create a different dataset split as already mentioned above, and fine tune on a completely imbalanced dataset. The first solution can be quite exhaustive, however, since the earth is a large place, and there will most likely always be places which are not included in the dataset. Therefore, the spatial bias of dam locations remains an unsolved problem.

8 Future work

This thesis proposed a first attempt at solving dam location with satellite imagery, and several things can be improved or researched. In terms of networks exploration, we have only explored a single YOLO network, but other methods may offer better performance, such as RCNN architectures. Another tweak that can be experimented with is directly using low level feature maps for prediction. This has proven to increase localization accuracy in other detection problems, and could be helpful for this problem as well. Another promising area of improvement is to use Generalized IOU (GIOU) over the regular IOU that was used throughout this thesis in the loss function. The GIOU provides more stability during training if it is used in the loss functions, as it gives a sense of direction in which the model should improve. Naturally, other feature sets can be explored as well, inlcuding adding extra short wave infrared bands directly, of which most of the water index calculations are made up of. That way, the network is forced to create its own water feature itself.

Another problem was the amount of data that is needed to obtain enough spatial coverage for the entire globe. An easy solution is to simply sample more data, but this will pose storage issues, especially with extra feature bands. An interesting approach to try is using Generative Adverserial Networks (GAN) to aid in this endeavour, along with data augmentation. These networks can, if fine tuned correctly, create fairly realistic artificial samples of the object in question. In our case, they could generate artifical dams or sceneries without dam locations, which can aid in training dam classifiers, alongside data augmentation.

Finally, we have used a sliding window with overlap to analyze large areas. The added overlap was primarily needed because the patches were often too small to ensure that a dam was fully emerged in it, which can result in a single dam being divided over multiple patches, making the dam unrecognizable. As a solution, larger input image patches can be tried

9 Conclusion

In this thesis, we investigated whether it was possible to locate and count dams using satellite data. The proposed methodology consisted of two steps consisting of first finetuning a backend network, which was then transferred to a YOLO v2 object detector. Finally, the two steps were then combined and tested on large input images, adapting it to a more realistic scenario. In conclusion, this thesis aimed to provide evidence for several questions, which we will now try to answer:

- RQ1: Object detection algorithms and CNN architectures can indeed be used to classify and localize dam locations using satellite data. We think that the achieved scores provide enough evidence to back up this claim.
- RQ2: Different input feature representations definitely do increase model performance. The input feature representation using all 5 features significantly outperformed the baseline, consisting of only RGB channels. The only sidenote is that elevation features sometimes seem to slightly reduce the results in some cases, which might be due to the difference in its initial resolution.
- RQ4: We feel we can at least partially answer this question. Large scale areas have added difficulty that they become harder to compute due to time constraints and memory issues, and models that are trained on smaller patches do not necessarily generalize well to larger input sizes. This thesis and earlier work [20] has shown that it is at least computationally possible to analyze large areas of virtually any size, using models that are trained on smaller patches. However, the detection results still vary from region to region.
- RQ3: We do not think we have (fully) solved the spatial bias issue. Despite sampling a large number of regions for training, and involving many dam locations, there are still regions in which the detection results are poor, since they generate many false positives.

References

Articles

[1] Günther Grill et al. "An index-based framework for assessing patterns and trends in river fragmentation and flow regulation by global dams at multiple scales". In: *Environmental Research Letters* 10 (Jan. 2015), p. 015001. DOI: 10.1088/1748-9326/10/1/015001.

[2] C. Vörösmarty et al. "Global threats to human water security and river biodiversity". In: *Nature* 467 (Sept. 2010), pp. 555–561. DOI: 10.1038/nature09440.

[3] Edward Anthony et al. "Linking rapid erosion of the Mekong River delta to human activities". In: *Scientific Reports* 5 (Oct. 2015), p. 14745. DOI: 10.1038/srep14745.

[4] Bernhard Lehner et al. "High-resolution mapping of the world's reservoirs and dams for sustainable river-flow management". In: *Frontiers in Ecology and the Environment* 9.9 (2011), pp. 494–502. DOI: 10.1890/100125. eprint: https://esajournals.onlinelibrary.wiley.com/doi/pdf/10.1890/100125. URL: https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1890/100125.

[6] Dominik Wisser et al. "Reconstructing 20th century global hydrography: A contribution to the Global Terrestrial Network- Hydrology (GTN-H)". In: *Hydrology and Earth System Sciences* 14 (Jan. 2010). DOI: 10.5194/hessd-6-2679-2009.

[7] Johan R Meijer et al. "Global patterns of current and future road infrastructure". In: *Environmental Research Letters* 13.6 (May 2018), p. 064006. DOI: 10.1088/1748-9326/aabd42. URL: https://doi.org/10.1088%2F1748-9326%2Faabd42.

[8] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *IEEE International Conference on Computer Vision (ICCV 2015)* 1502 (Feb. 2015). DOI: 10.1109/ICCV.2015.123.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: 10.1145/3065386.

[10] J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[11] J. Redmon and A. Farhadi. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[12] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *ArXiv* abs/1804.02767 (2018).

[13] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 91–99.

[14] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *ECCV*. 2016.

[15] Emilio Guirado et al. “Whale counting in satellite and aerial images with deep learning”. In: *Scientific Reports*. 2019.

[16] Z. Liu et al. “Rotated region based CNN for ship detection”. In: *2017 IEEE International Conference on Image Processing (ICIP)*. Sept. 2017, pp. 900–904. DOI: 10.1109/ICIP.2017.8296411.

[17] H. Lechgar, H. Bekkar, and H. Rhinane. “DETECTION OF CITIES VEHICLE FLEET USING YOLO V2 AND AERIAL IMAGES”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-4/W12 (2019), pp. 121–126. DOI: 10.5194/isprs-archives-XLII-4-W12-121-2019. URL: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-4-W12/121/2019/.

[18] Junyan Lu et al. “A Vehicle Detection Method for Aerial Image Based on YOLO”. In: *Journal of Computer and Communications* 06 (Jan. 2018), pp. 98–107. DOI: 10.4236/jcc.2018.611009.

[19] Yang-Lang Chang et al. “Ship detection based on YOLOv2 for SAR imagery”. In: *Remote Sensing* 11 (Apr. 2019), p. 786. DOI: 10.3390/rs11070786.

[20] Emilio Guirado et al. “Automatic whale counting in satellite images with deep learning”. In: (Oct. 2018). DOI: 10.1101/443671.

[21] T. Ishii et al. “Detection by classification of buildings in multispectral satellite imagery”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. Dec. 2016, pp. 3344–3349. DOI: 10.1109/ICPR.2016.7900150.

[23] Recording, Coding Guide for the Structure Inventory, and Appraisal of the Nation’s Bridges. “Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation’s Bridges”. In: Report No. FHWA-PD-96-001 (1995).

[26] J.-F Pekel et al. “High-resolution mapping of global surface water and its long-term changes”. In: *Nature* 540 (Dec. 2016). DOI: 10.1038/nature20584.

[28] 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.

[29] K. He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[30] G. Huang et al. “Densely Connected Convolutional Networks”. In: (July 2017), pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.

[31] J. Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (Nov. 1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[32] S. K. McFEETERS. “The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features”. In: *International Journal of Remote Sensing* 17.7 (1996), pp. 1425–1432. DOI: 10.1080/01431169608948714. eprint: https://doi.org/10.1080/01431169608948714. URL: https://doi.org/10.1080/01431169608948714.

[33] Hanqiu Xu. “Modification of Normalized Difference Water Index (NDWI) to Enhance Open Water Features in Remotely Sensed Imagery”. In: *International Journal of Remote Sensing* 27 (July 2006), pp. 3025–3033. DOI: 10.1080/01431160600589179.

[34] R. R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

Websites

[5] M. Mulligan et al. *Global dams database and geowiki Version 1.* 2009. URL: http://geodata.policysupport.org/dams. (visited on 10/08/2019).

[22] Federal Highway Administration. *National Bridge Inventory.* 2019. URL: https://www.fhwa.dot.gov/bridge/nbi.cfm (visited on 09/01/2019).

[24] Earth Engine Data Catalog. *Sentinel-2 MSI: MultiSpectral Instrument, Level-1C.* 2019. URL: https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2 (visited on 10/15/2019).

[25] Earth Engine Data Catalog. *ALOS DSM: Global 30m.* 2015. URL: https://developers.google.com/earth-engine/datasets/catalog/JAXA_ALOS_AW3D30_V1_1 (visited on 10/15/2019).

[27] Earth Engine Data Catalog. *JRC Monthly Water History, v1.1.* 2016. URL: https://developers.google.com/earth-engine/datasets/catalog/JRC_GSW1_1_MonthlyHistory (visited on 10/15/2019).

10 Appendix

Possible other sections (appendix):

- Handling tfrecords? a nice read in an appendix on how this cost me a lot of time
- Same goes for handling and sampling through GEE

•

things I tried but did not work (or possible flaws of this research (data):

- segmentation maps: these were irrelevant after object detection became available
- 10 m resolution and (very) small dams.
- Grand is inaccurate
- GOOD has many 'insignificant' dams
- NID (american dam survery) is not good, since locations are often wrong
- high res images did not work
- s2/sr images did not work because they do not have enough data
- water mask vs ndwi: water masks are made out of ndwi like calculations
- Handling class imbalance: either via under/oversampling or class weights.
- Researching some way to optimize the neural network architecture in some guided way (e.g. via an optimization function).
- For evaluation, it would be better to have segmentation labels so that other metrics can be used as well.
- Testing the effect of adding a third class for bridges, since they could be the cause for a substantial portion of false positives.
- Find a way to use segmentation models, despite only having classification labels.

10.1 Data discrepancies

During data extraction several problems came to light that might afflict the entire pipeline. The first problem is that the distance between the annotated dam location and the actual dam location seems to be substantial in several samples. With substantial we mean that the distance is larger than 2 km. In the current pipeline implementation with spatial a resolution of 10 meters, this means that the image patches have to be greater or equal than 201x201 in order to include the actual dam. An example is shown in figure 14.

Figure 14: The dam location is far away from the actual dam near the bridge at the bottom. The distance is approximately 2 km.

The second problem is that the dams in the GOOD dataset are sometimes insignificantly small, and there might be discussion whether it is a dam at all or not. They can occur for example near farm land in a small pond or creek. At this time, it is impossible to filter out all of the possible dams that are of questionable size. This is shown in figure 15.

Figure 15: Location of a questionable dam.

A final hurdle is that spatial resolution of the S2 satellite, which is sampled at 10 meters, and currently the best freely available. Although the current resolution is fine for most large dams, it might not be sufficient to classify smaller dams, as they would be only a single pixel, or would not be visualized on the image grid at all. In order to overcome this, there do exist several high resolution images in the Earth Engine catalogue, but they do not span the entire globe, and have a lower time coverage than the S2 data.