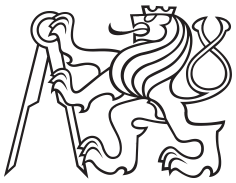
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Cloud and Shadow Detection in Satellite Imagery

Bc. Matěj Bartoš

Supervisor: Ing. Tomáš Pajdla PhD.
Field of study: Open Informatics
Subfield: Computer Vision and Image Processing
May 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Matěj B a r t o š

Study programme: Open Informatics

Specialisation: Computer Vision and Image Processing

Title of Diploma Thesis: Cloud and Shadow Detection in Satellite Imagery

Guidelines:

1. Review the state of the art in cloud and shadow detection in satellite imagery [1-6].
2. Suggest and develop a cloud and shadow detection method exploiting recent advances in CNN [6] technology. Consider the problem of training data construction and simulation.
3. Implement the method and demonstrate it on data from GISAT s.r.o. Provide an experimental evaluation of the performance of the new method in comparison to the state of the art method used by GISAT s.r.o.

Bibliography/Sources:

[1] Ben V. Hollingsworth and Liqiang Chen and Stephen E. Reichenbach and Richard R. Irish. "Automated cloud cover assessment for Landsat TM images". In: Proc. SPIE, Imaging Spectrometry II2819 (Nov. 1996), 170-181.
[2] K. He and J. Sun and X. Tang. "Single Image Haze Removal Using Dark Channel Prior". In: IEEE Transactions on Pattern Analysis and Ma-chine Intelligence33 (2011), pp. 2341-2353.
[3] Zhe Zhu and Curtis E. Woodcock. "Object-based cloud and cloud shadow detection in Landsat imagery". In: Remote Sensing of Environment118 (Mar. 2012), pp. 83-94.
[4] Q. Yuan and G. Yang X. Li and H. Shen and L. Zhang and H. Zhang. "Re-covering quantitative remote sensing products contaminated by thick clouds and shadows using multi-temporal dictionary learning". In: IEEE Transactions on Geoscience and Remote Sensing52.11 (Nov.2014).
[5] J. Wang and P. A. Olsen and A. R. Conn and A. C. Lozano. "Removing Clouds and Recovering Ground Observations in Satellite Image Sequences via Temporally Contiguous Robust Matrix Completion". CVPR 2016.
[6] LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning". Nature. 521 (7553): 436- 444.

Diploma Thesis Supervisor: Ing. Tomáš Pajdla, Ph.D.

Valid until: the end of the summer semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 6, 2017

Acknowledgements

I would like to thank Ing. Tomáš Pajdla PhD., for his advices, support and patience throughout the thesis work, the Center for Machine Perception for computer time and resources, Lucie Černá for grammar corrections and styling.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

Prague, date

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne

Abstract

In recent years there has been an enormous growth in the amount of publicly available satellite imagery and overall satellites launched, which has imposed a challenging data problem of how to label or classify objects on satellite imagery. This thesis reviews Fmask algorithm[1], a state of the art solution, of cloud and shadow detection, and explores a problem of synthesizing satellite data and a problem of semantic labelling of satellite imagery by designing, implementing and evaluating neural network. The resulting pipeline synthesizes image into clouds and background. The modelled clouds can be then combined with any other image creating a new or enhanced data. The main contribution of this thesis is the utilization of the dataset synthesis in learning of neural networks. We have achieved 94.3% accuracy on a real world dataset. Neural networks were created with a help of Caffe framework [2].

Keywords: Meta-ball, Landsat, Caffe, Hluboké učení (Deep learning), SegNet, Satellite Imagery

Supervisor: Ing. Tomáš Pajdla PhD.

Abstrakt

V posledních letech došlo k enormnímu nárůstu veřejně dostupných satelitních snímků a celkového množství vynešených satelitů, což předložilo náročný problém s daty, jak označit nebo klasifikovat objekty na satelitních snímcích. Tato práce uvede algoritmus Fmask[1], state of the art řešení, detekce mraků a stínů, a zkoumá problém syntézy družicových dat a problém sémantického značení družicových snímků návrhem, provedením a vyhodnocením neuronové sítě. Výsledný algoritmus syntetizuje obraz na oblaka a zem, které lze kombinovat s jakýmkoliv jiným obrázkem, a tím se vytvoří nová nebo vylepšená stávající data. Hlavním přínosem této diplomové práce je využití syntézy datasetu při učení neuronových sítí. Na skutečném datasetu jsme dosáhli 94.3% přesnosti (accuracy). Neuronové sítě byly vytvořeny za pomoci knihovny Caffe[2].

Klíčová slova: Meta-ball, Landsat, Caffe, Deep Learning, SegNet, Satelitní snímky

Překlad názvu: Detekce mraků v satelitních obrazech

Contents

Figures

Tables

Chapter 1

Introduction

Earth observation (EO) is a process of gathering the information about planet Earth through remote sensing. The location, where we can gather the most information about our planet, is in space. Earth Observation data has many usages e.g. forecasting weather, urban growth studies, biodiversity and wildlife studies.

The Electromagnetic (EM) spectrum is important because each object reflects, transmits and absorbs light differently, depending on its chemical composition. The objects reflect light in bands of light we cannot see with our eyes – but sensors can. The spectrometer records the light that the objects reflect into bands. The plants are colored green because they reflect more green light. Healthy vegetation reflects more near-infrared light and we use an index called Normalized Difference Vegetation Index (NDVI) to help classify the vegetation. Each object has its own unique chemical composition. This is in an accordance with saying that each object has its own spectral signature. The differences in spectral signatures can be used to distinguish the objects apart. The spectral signatures in the EM spectrum give us the ability to learn more information about Earth's features that we may not have known otherwise. [3]

There are two branches in remote sensing: active sensing and passive sensing. The passive sensors measure reflected light emitted from the sun, whereas the active sensors have their own light source. There are electromagnetic waves coming from the sun hitting Earth on a daily basis, which enables the usage of the passive sensing. The most known representative of the active sensing branch is RadarSat-1 [4] and RadarSat-2 [5] which are still at service. The

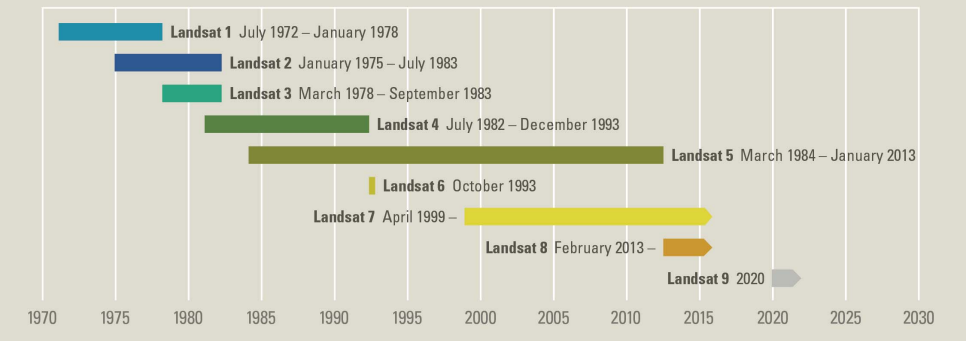


Figure 1.1: The Landsat Missions Timeline, with dates of launched satellites and dates of deactivations [7].

The Landsat satellite mission	Number of spectral bands	Resolution [meters]
Landsat 1	4	79 x 57
Landsat 2	4	79 x 57
Landsat 3	4	79 x 57
Landsat 4	7	30 x 30
Landsat 5	7	30 x 30
~~Landsat 6~~	~~8~~	~~30 x 30 *~~
Landsat 7	8	30 x 30*
Landsat 8	11	30 x 30*

* Panchromatic band has 15 x 15 resolution

Table 1.1: Specifications of Landsat satellites

main advantage of the active sensing over the passive sensing is the ability to collect images over night and day. The model example for the passive sensing is a Landsat project. The Landsat project represents the world's longest ongoing acquired collection of space-based moderate-resolution land remote sensing data. [6] The Landsat project is governed by a joint initiative between the U.S. Geological Survey (USGS) and NASA.

The Landsat project is an archive of satellite images. The Landsat project ensures continuity, in such way that the new satellites are planned to be launched in the ongoing years, even though the first satellite was launched in 1972. The new satellite *Landsat*9 is scheduled to be launched in 2020. The timeline of orbiting satellites and launch dates of Landsat satellites can be seen on figure 1.1.

1.1 Motivation

Prior to this thesis, there was a collaboration between Gisat s.r.o. and Czech Technical University In Prague (CVUT) on Urban Dynamic Processor [8], which brought up the importance of cloud and shadow detection. The Urban Dynamic Processor required low cloud coverage images for creating temporal sequences to classify a development of the urban coverage, agricultural coverage, forest cover etc.

We aim at developing a new algorithm for cloud and shadow detection based on Convolutional Neural Networks (CNN), which is a promising approach experiencing rapid development. The main reason for low quality of the neural network output, or even inability to properly learn a neural network, lies in the dataset constraints. Nowadays, there are many publicly available datasets that can be used as a benchmark for types and structures of the neural networks. However, they are very specific. For instance, ImageNet [9] is an excellent dataset for day to day images. Nevertheless the results on such a learned dataset would not obviously generalize properly on the satellite data. In such cases one can create his own dataset based on images given by a supervisor of the task. The problem is that a small number of training images used to represent the problem does not suffice to train a network to generalize sufficiently.

The problem becomes even worse if a ground truth labelling is mismatched for the actual truth data. The Fmask algorithm is the state of the art solution for the cloud detection in Gisat s.r.o., although it has great results, they can not be taken as a 100% ground truth, due to the visible mismatches; e.g. Fmask predicts a cloud pixel for a highly reflective roof. Fmask is a two pass algorithm for the cloud detection. The first pass consists of conditions in bands of LandSat images using for example NDVI, NDSI (Normalized Difference Snow Index), etc. The latter pass computes the probability of a cloud from the potential cloud pixels based on many thresholds on bands. The shadow of clouds is based on filtering in band 4 (Near InfraRed band), where the Flood fill algorithm is applied, by knowing the apriori sun position (zenith angle, azimuth angle) and satellite view angle, the matching of shadow and potential cloud is done. In addition, based on Ing. Tomáš Pajdla PhD. experiments with manual labeling of satellite imagery, there were discrepancies amongst same images labeled by different people, which implied the impossibility of the manual 100% accurate ground truth labelling [8]. The examples of a visible mismatch of the assigned labels by "ground truth" Fmask evaluation can be seen on a figure 1.2 or 1.3, where in reality such situations as a shadow without a cloud, can not happen.

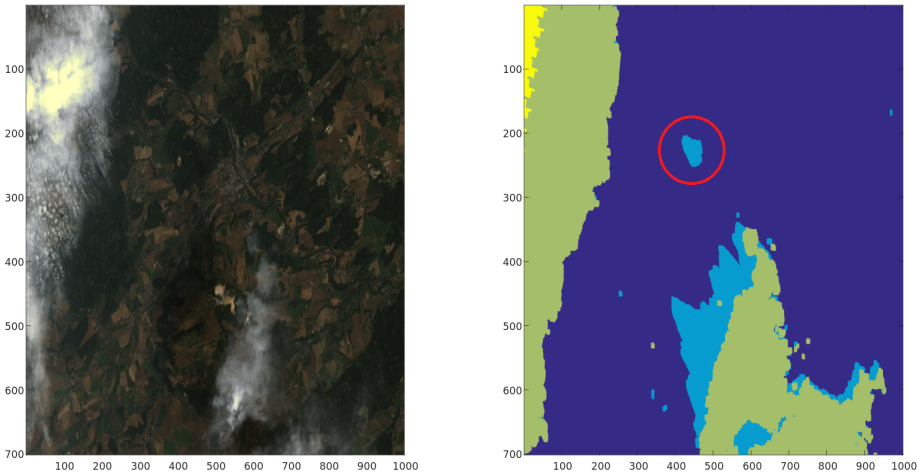


Figure 1.2: A ground truth assignment mismatch of a shadow. On the left is an RGB representation of the multichannel satellite image, on the right is the ground truth assignment. The yellow pixels represent the cloud, green stand for the shadow and blue for the background.

1.2 Data available

We were provided satellite data by the Gisat s.r.o. for developing cloud and shadow classifier. We have got 39 Landsat 6-channel tiff files, where the image pixels correspond to a calibrated reflectance in a range from 0 to 10,000, other values are invalid (over saturation, improperly calibrated or incorrect calibration parameters, etc.). The interesting region of the images was in the middle of the image surrounded by "no observation" label. The images were cropped to include as little "no observation" label as possible, which have left them of about 8000x6000 resolution. However we have acquired old images used in Urban Dynamic Processor from previous work with Gisat s.r.o. [8], which had different label coding and different resolution, specifically 2500x2000, otherwise they were similar except of mean brightness. We divided these values in subsequent work by a potential maximal value, 10,000, converted data to float and scaled them by 255 and converted to uint8. The 6 channels are identical to the first 6 channels of images taken by Landsat 4 to Landsat 8. We preprocessed the new images, since each band of the 6 channels was in a separate file, which was cumbersome. We therefore merged all band files to one representing all bands. A metadata layer, an image of the same size containing a class assignment, is assigned for every image. The categories or labels contained in metadata layers:

0 clear land pixel

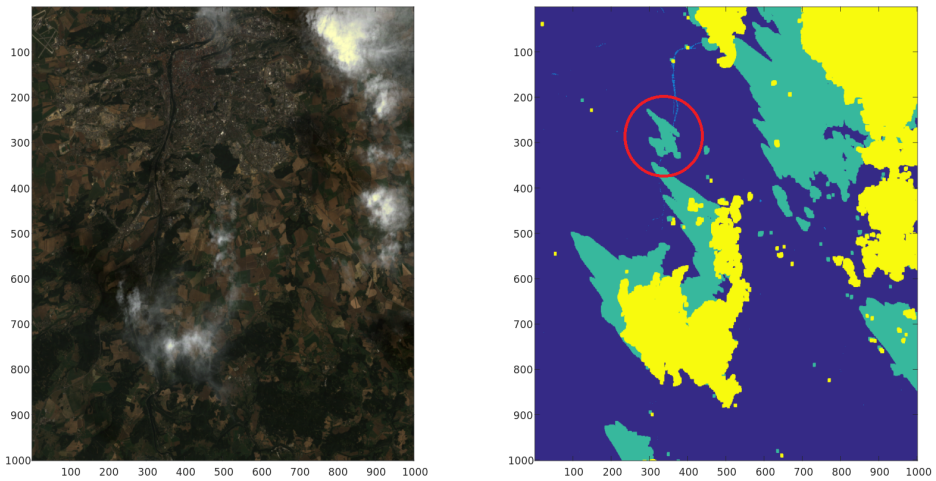


Figure 1.3: A ground truth assignment mismatch of a shadow. On the left is an RGB representation of the multichannel satellite image, on the right is the ground truth assignment. The blue pixels represent the background, light blue stand for the shadow and yellow-brown are the cloud pixels.

1 clear water pixel

2 cloud shadow

3 snow

4 cloud

255 no observation

The snow class/label was omitted and the clear land pixel was merged with the clear water pixel to a new class "background" in all the following work for an easier evaluation and not being the part of a research. The new resulting classes are as follows:

0 background

1 cloud shadow

2 cloud

3 no observation

There are other sources for obtaining satellite data, mainly publicly available Landsat satellite images, through [10], [11], [12]. After registering for free, we can possibly download all images archived, even with metadata included as a sun elevation, sun azimuth, acquisition date, etc. On the other hand

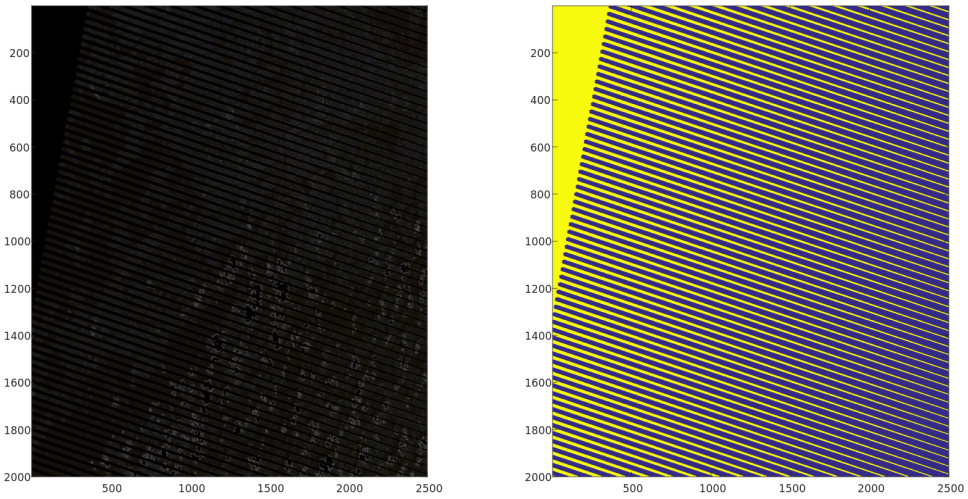


Figure 1.4: The results of a faulty sensor. On the left is an RGB representation of multichannel satellite image, on the right is the ground truth assignment. Blue represents the background pixels, whereas yellow stands for the no observation label.

these web-pages do not include mask of classes, however the images can be evaluated on the resulting neural network and visually compared, but they can not be part of the training process.

The main problem with our data as it was stated in section 1.1 was a visible imperfect match in the data and the assessed label evaluation, that led us to a theory of synthesizing our own cloud data and creating a proper label evaluation.

1.3 State of the Art

The haze reduction algorithm was developed in [13], which performs thresholding on a dark channel, the minimum pixel intensity values across the RGB channel, based on the assumption that patches of an image have a constant transmission. Originally it was thought that this algorithm can be used as a low cloud detector, nevertheless performed poorly on satellite data. On the other hand the developed algorithm works great on outdoor hazed images.

A new algorithm called Tmask (multiTemporal mask) developed in [14] for automated masking of cloud, cloud shadow, and snow for multitemporal Landsat images. This algorithm consists of two steps: sieve most of the clouds, cloud shadows and snow with Fmask algorithm. The second step

uses a Robust Iteratively Reweighted Least Squares method to estimate a time series model for each pixel by comparing model estimates with Landsat observations. A snow threshold is derived for Band 5 reflectance for each pixel at each specific time based on a modified Norwegian Linear Reflectance-to-Snow-Cover algorithm.

In this paper they improved preceding algorithm Fmask: improvements in the Fmask program for Landsat 4–7, a new version which takes an advantage of the new cirrus band in Landsat 8 and a prototype algorithm for Sentinel 2 images. Even though Sentinel 2 images do not have a thermal band to help with cloud detection, the new cirrus band is found to be useful for detecting clouds, especially for thin cirrus clouds. For Landsat 8, almost all the Fmask algorithm components are the same as for Landsat 4–7, except a new cirrus cloud probability is calculated using the new cirrus band, which improves the detection of thin cirrus clouds. Landsat 8 results are better than the Sentinel 2 scenario, with 6 out of 7 test images showing higher accuracies.[15]

In many papers they prove that the Normalize Difference Vegetation Index [16] can be used to detect vegetation on satellite images. In addition, the NDVI can be used as an indicator for a drought.

A method was proposed in [17] to co-train two dictionary pairs, one pair generated from the high resolution image (HRI) and the low resolution image (LRI) gradient patches, and the other generated from the HRI and synthetic-aperture radar (SAR) gradient patches. It was demonstrated that such combination of multiple data types improves the reconstruction results, as it is able to provide both low and high-frequency information.

The approach in [18] first detects and then removes the cloud-contaminated part of the image sequences based on [13], followed by post-processing to distinguish between a stationary white background and white clouds. Then it recovers the missing scenes from the clean parts using proposed TECROMAC objective. The objective function balances the temporal smoothness with a low rank solution while staying close to the original observations.

The article introduces an upgrade on the ACCA algorithm for classifying Landsat images by percentage of the estimated overall cloud cover to cloudiness categories, however this algorithm is currently obsolete [19].

They propose a new method for modelling clouds from a single photograph in [20]. Their method is capable of synthesizing three types of clouds: cirrus, altocumulus, and cumulus. They use a different representation for each type:

two-dimensional texture, meta-balls, volume data. Unfortunately their focus is primarily on images captured from the ground and their method also needs apriori camera calibration and ground truth of the background color or the possibility to create a new background with smoothing using the Poisson equation.

There is a method that represents cloud as a structured particle system, which describes macroscopic characteristics and the series of unstructured particles system describes volumetric detail of a cloud [21].

They have incorporated convolutional networks into a semantic labelling of satellite images in [22]. They adopt recently proposed architectures CaffeNet and GoogLeNet and resort to comparison of learning CNNs from scratch and fine-tune them. The experimental results have been shown on two datasets, namely UC-Merced, which consists of the aerial optical images with low-level characteristics similar to those of the Imagenet, and a Brazilian coffee dataset, which was released in 2015 and includes scenes taken by the SPOT sensor in the green, red, and near-infrared bands, over four counties in the State of Minas Gerais, Brazil. They showed that for the UC-Merced dataset the best result was a fine-tuned GoogLeNet architecture for 20,000 iterations and achieving 97.10% accuracy, whereas on Brazilian coffee dataset they proposed a GoogLeNet trained from scratch with 91.8% accuracy.

The more thorough comparison of architectures, datasets and their accuracy can be found in [23], where they compared three datasets: UC-Merced, RS19 and Brazilian Coffee datasets with 6 architectures: PatreoNet, AlexNet, CaffeNet, GoogLeNet, VGG ConvNets and OverFeat ConvNets. They also compared them with following strategies: fully training, fine-tuning and convolutional networks as feature extractors. The results point that the fine-tuning tends to be the best performing strategy. In fact, using the features from the fine-tuned ConvNet with linear Support Vector Machines (SVM) obtains the best results.

The novel strategy to eliminate main drawback of classification of large images with a sliding window approach can be found in [24]. They propose a scheme to reduce the classification time through the usage of superpixels. They have experimented with the different sized patches of superpixels and their affect on the classification of the Convolutional neural networks.

Chapter 2

Satellite imagery synthesis

We present a solution for a situation where there is erroneous ground truth, by artificially generating new images based on a given domain. We work with Landsat images in this thesis for which we present a "cloud synthesizer". The resulting synthesizer can generate an arbitrary amount of artificial images given a cloud free image and a reference cloud covered mask and an image, from which the clouds will be modelled. The synthesizing and the modelling algorithms were produced in Matlab.

2.1 Modelling

We found a cloud-free image in a given batch of Landsat images, which in this chapter will be referred to as a ground truth. We perform the modelling of clouds for a particular cloud-covered image and mask, and the output of the algorithm would be synthesized modelled clouds.

We were given metadata for each cloud image, consisting of a scene identification, acquisition data, sun azimuth and sun elevation. It is possible to trace them back to the original images on [10], [11], [12]. The desired metadata input can be guessed or appropriately set by a similar image even without metadata.

The reference cloud covered mask is preprocessed with eroding, since we

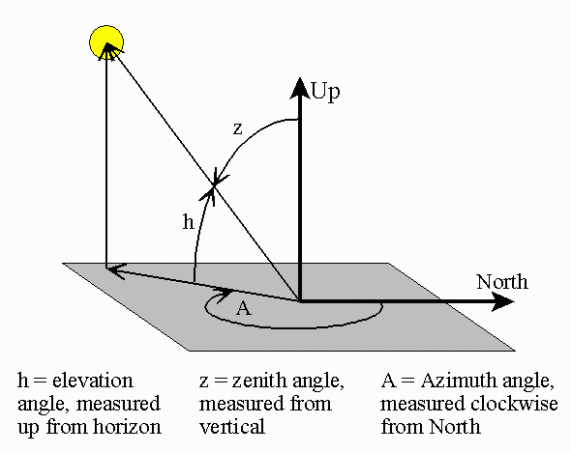


Figure 2.1: The correspondences between the sun elevation, sun azimuth and sun zenith [25].

want to get rid of potentially lonely pixels like roofs, that are probably not clouds and we do not want to model small clouds on the edge of the cloud region. Therefore the mask is eroded with the disc structural element of a size 15. Then we cross-erase failures of both masks so the ground truth would be in the same poor situation as is the reference cloud mask. Subsequently we remove supposedly non cloud pixels from masks. The previous step was primarily intended for such images that have failures in sensors resulting in diagonals of pixel of "no observation" (figure 1.4) or even whole bands of such case.

We compute a sun zenith angle, an angle from y-plane, from the sun elevation, which is an angle from x-plane, based on basic trigonometry as can be seen on a figure 2.1, that will be referred to $angle_Y$.

$$angle_Y = \frac{90 - sunelevation}{180} * \pi$$

There were two options for creating the artificial clouds:

1. Generate perlin noise as cloud cover.
2. Approximate real clouds with blobs (meta-balls).

We concluded that the meta-balls are better approximation of real life clouds than perlin noise after experimenting with meta-balls. The method for modeling three-dimensional clouds based on a satellite image is proposed

in [26]. The meta-balls represent the density distribution of clouds. The parameters of meta-balls (center positions, effective radii, and density values) are determined automatically so that a synthesized image of the clouds coincides with the satellite image. Concerning the satellite image, the proposed method makes use of the fact that the color of clouds can be calculated by integrating the scattered light due to the particles in them. Clouds with the shape and color similar to real clouds can be automatically generated in this method. However this method does not reconstruct the exact three-dimensional shape of clouds in the satellite image.

Determining parameters of the meta-balls is equivalent to solving an inverse problem of determining the density distribution inside the clouds so that the image of synthesized clouds is similar to the satellite image. As stated earlier, the problem is very complicated and hard to solve. Therefore, they assumed that the multiple scattering can be neglected. Furthermore the attenuation of light due to the cloud particles is approximated by a constant. Despite these assumptions, there is no unique solution for the problem. Therefore, the parameters of the meta-balls are heuristically determined as follows. First, each pixel of the satellite image is classified as the part of either a cloud region or a background region. The satellite image is converted to monotone to accomplish that. Then the pixels with intensities higher than a specified threshold are identified as clouds. Next, one meta-ball is added at the pixel with the maximum intensity in the cloud region. After that, its radius and density at the center are optimized and the approximated image is calculated from the clouds modeled by meta-balls. Then a new meta-ball is added if the error between the satellite image and the approximated image is higher than a specified threshold. These processes are repeated until the error is lower than the threshold. [26]

Meta-balls are characterized by a field function, in this thesis we followed [26] and [27].

The field function of the meta-ball has a following equation:

$$f(r,R) = -\frac{4}{9}\left(\frac{r}{R}\right)^6 + \frac{17}{9}\left(\frac{r}{R}\right)^4 - \frac{22}{9}\left(\frac{r}{R}\right)^2 + 1,$$

where r means a distance from a center of the meta-ball and a pixel we are currently working with.R stands for a radius of the meta-ball or in this sense a radius of a cloud. When $r > R$, then the whole equation evaluate to 0. After computing $f(r,R)$ for every meta-ball given one pixel, there is

computed a cumulative density of meta-balls for the one pixel P as:

$$\rho(P) = \sum_{j=1}^{N} q_j f(r_j, R_j),$$

where q_j is a density of j-th meta-ball. The cumulative density represents the sum of portions of densities from all meta-balls, whose ranges occupy a given pixel. This density is then multiplied by constants, in such matter to mirror physical aspects, and added to ground truth pixel, and in result representing the intensity of a cloud.

The algorithm for generating one meta-ball can be briefly described as follows:

1. Find the center for a new meta-ball.
2. Initialize the density and the radius of the meta-ball.
3. Optimize the density and the radius to better fit the data.
4. Remove the optimized meta-ball pixels from the reference image.
5. (Stabilize solution.)

We have attempted to lower down the number of channels in an input image to reduce the dimensionality of a problem, in such way that the input image would be transformed to a gray-scale. However, a question arose of how to properly set densities for all the channels and also this attempt also proved of very poor approximation of the problem, because of the huge mismatches in choosing the center of the meta-ball. Thus, we have settled on original multichannel input images.

This algorithm loops itself until there is a maximal given number of meta-balls or a stopping criterion for finding new center is reached.

Initially it was thought that for finding the new coordinates for the center a maximum of the sum of squared differences (SSD) of the ground truth and the reference image would be a great metric to recognize clouds from the background. However it was determined after the experiments with this metric, that to better discriminate clouds from the background a maximum of the sum of absolute differences is outperforming the SSD. The summation part of the metric is made across the channels of the image. We compute the

subscripts (row and column indices) of a maximal value of a given metric:

$$\arg\max_{row,col} f(row,col) = \left\{ \sum_{c=0}^{N} |Image_{ref}(row,col,c) - Image_{gt}(row,col,c)| \right\},$$

where N stands for a number of channels and the rest is self explanatory.

Then we compute an initial density of the meta-ball:

$$initialdensity = \frac{Image_{ref}(row,col,:) - k_1 Image_{gt}(row,col,:)}{(2 * SunIntensity * angle_Y * k_2)},$$

where $SunIntesity$ represents an intensity of the sunlight outside the atmosphere. We substituted the intensity with a vector of value 255 with the same size as is the number of the channels of an original image. k_1 and k_2 are constants, which approximate the real physical equation [26]. $angle_Y$ is an earlier mentioned sun zenith angle.

There is a need to give an initial radius to our optimization to acquire a better model of a cloud. The first method was a fixed initial radius that will potentially grow in the optimization process, nevertheless the modelled clouds have not have large radii, which defied the assumption that large clouds should have large radii. Instead the optimization generated a huge amount of a small radius meta-balls. The second method was to begin with large radius for the initialization and as the amount of meta-balls grows the initial radius decreases. This method outperformed the former one, but did not worked well when there was a mismatch with the large radius. The last method for the initialization was to create a probability function that will determine the initial radius based on a random sample. This method performed the fastest convergence on the given data and thus fastest optimization, without a loss of much accuracy.

This step is followed by the optimization which takes the center of a meta-ball, given by a maximum of metric, initial density and initial radius. We need to optimize the density and the radius. Originally we optimized the density and the radius with Matlab fminsearch, which finds a minimum of unconstrained multi-variable function using the derivative-free method. However the optimization took a high amount of time. The main problem was that the optimization of the density was made in all channels. There were 7 variables (6 densities and a radius) in our case. Then we rejected fminsearch and created a discrete stepping, which evaluated the initial configuration and calculated the resulting values for each step until the stopping condition was met. The step is mentioned here in sense of $radius_n = radius_{n-1} + step_{radius}$ or $density_n = density_{n-1} + step_{density}$. The optimization is stopped when an optimized value is not decreasing or the number of iterations is met. The

latest method for the fastest computing without losing much of the modelling accuracy is divided into two related components. In the first part we optimize only the radius, in which is the optimization of the density. We rejected the necessity of minimizing the density across all channels in the optimization of the density, since after thorough observation we find out that the 6 optimized densities did not differ much. Instead of it we optimized one variable, which represented all channels at once, and the idea proved a massive speedup, however lost some of the modelling accuracy.

There is also an approximation in the optimization function for a further speedup in such way, that the optimization function does not compute the cumulative density across all pixels with all meta-balls for each step again, however there is always only one meta-ball being optimized. If one meta-ball is already optimized then its cloud image is added to the ground truth image. There is no need to recompute all meta-balls at once after this approximation.

The optimization function is simplified in the same way to generate the cumulative density only in the circumscribed square, there is no need to generate it for all pixels.

We need to remove generated clouds of the given meta-ball from the reference image, after all the optimization is done and ground truth is updated, to prevent modelling of the same clouds and to prevent choosing a new meta-ball in a close neighbourhood.

Even though removing the generated clouds should work out of the box, there is a need to stabilize the solution, because of the inaccuracies in the process. We need to set the center pixel of the meta-ball of the ground truth to the same value as is in the reference image to avoid choosing the same center for new meta-balls to stabilize the solution.

Eventually the meta-ball structure contains a center of the meta-ball, the density, that is the same for every channel, and the radius. We saved the precomputed bitmap of the cumulative densities in the circumscribed square of the radius in the optimization process to this structure also, to prevent recomputing of the added value of meta-ball cloud. The only computation needed to get a real cloud value was to multiply this bitmap by the sun zenith ($angle_Y$), constant k_2 and the sun value (255).

The summary of this procedure can be seen in the pseudo-code 1.

Result: Modelled cloud
Preprocessing - erosion of reference image;
Initialize;
while *num_of_metaballs <= max_of_metaballs* **do**
    Find center and maximum from $|Image_{ref} - Image_{gt}|$;
    if *maximum <= stopping_value* **then**
        STOP;
    end
    Initialize the density and the radius;
    metaball = Minimize_metaball($Image_{gt}$, $Image_{ref}$, center, density, radius));
    Generate_cloud_image($CroppedImage_{gt}$, metaball);
    Add the clouds to $Image_{gt}$;
    Remove the clouds from $Image_{ref}$;
    Set the value of $Image_{gt}$(center) to $Image_{ref}$(center);
    Add the metaball to the collection and increment num_meta;
end

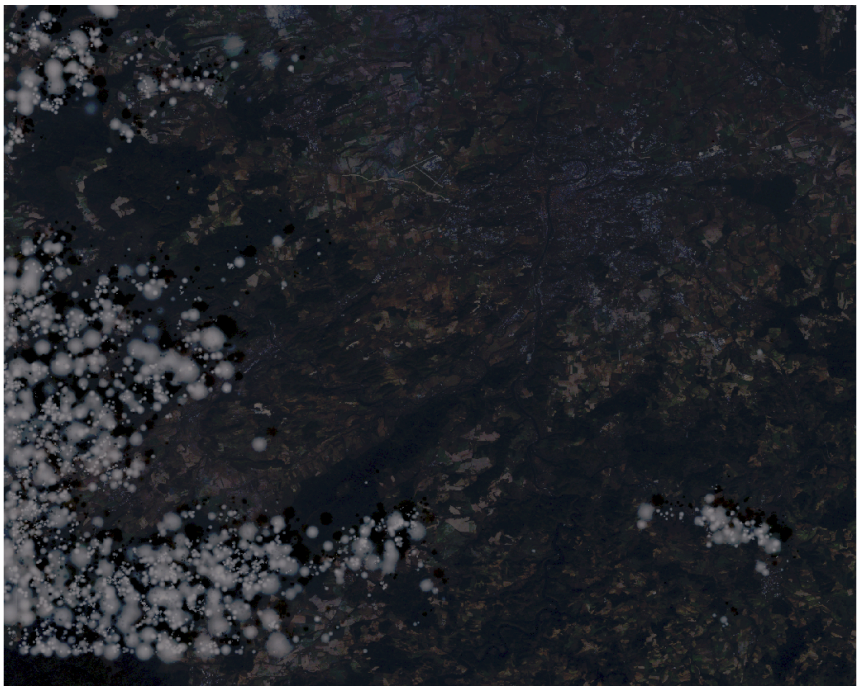
Algorithm 1: Cloud Modelling

2.2 Synthesizing

The modelled clouds are placed on ground truth image based on two options. The former option stands for a randomly placed patch of clouds, whereas the latter option is moving all clouds simultaneously by a given offset. There is a classical k-means algorithm performed on the meta-balls centers for the random patches with one quarter of a number of clouds as the number for centers. Then the clouds assigned to a given center are shifted by a random offset from their previous locations in both directions and placed on the ground truth image. Concerning the second option, all clouds are moved from their location by a certain offset. All satellite images were visualized in an RGB format in such way that the first three bands were rearranged to fit the RGB needs.

Each meta-ball is given a new center by one of the previously mentioned methods, which is just the addition of a constant to the old coordinates. Some of the meta-balls can wind up out of the image coordinates, so we need to leave them out. We generate a random height of the cloud for each meta-ball in a range from the minimal cloud height to the maximal cloud height set in the configuration. We compute a new center for a "shadow meta-ball" from the given sun azimuth, cloud height and meta-ball center, that is the same as the meta-ball, nevertheless in the end it will be subtracted instead of added to the final image.

(a) : An image synthesized by the clustering option



(b) : An image synthesized by the moving option

Figure 2.2: The differences in synthesizing options, the images were visually enhanced

$$increment = \tan(sunzenith) \cdot cloudheight$$

$$direction = [\cos(sunazimuth), \sin(sunazimuth)]$$
$$delta = increment \cdot direction$$
$$x_{new} = x_{old} + delta_x$$
$$y_{new} = y_{old} + delta_y$$

We initialized a new image cloud bitmap with the same size as the image and for each meta-ball we appended its cloud bitmap to this new-image. The same procedure will be applied for shadows.

There was an effort to make the clouds boundaries less visible, so we tried to process our image cloud bitmap through Gaussian filter. The images were aesthetically pleasing after the procedure, nonetheless for the later classification there were more cloud pixels around the edges in masks, where it was not possible to distinguish them from the background by eye.

The placing of a cloud/meta-ball is carried out by a following procedure:

1. Define a cloud map and a shadow map, in which all the resulting coefficients will be appended
2. Check if the cloud is out of bounds, eventually crop.
3. Get a random integer in a given range representing a cloud height.
4. Based on the sun azimuth and the cloud height cast a shadow.
5. Compute an actual brightness coefficients of a cloud from a precomputed bitmap with:

$$bitmap_{cloud} = k_2 * angle_Y * precomputed\_bitmap$$

6. Append $bitmap_{cloud}$ to the cloud map and with same coefficients also into the shadow map.

We transform the shadow map into the real brightness values by multiplying every element by maximal value 255 and transforming to uint8, which resolve into the new variable map_{shadow} after the placing procedure is done for each meta-ball. The same transformation process will be made for the cloud map, however we introduce a new variable - transformed cloud map as $output\_map_{cloud}$ and the old coefficients before the multiplication are in map_{cloud}. The merging of background, the cloud map and the cloud shadow is as follows:

1. $$\alpha = 0.85,$$

2. $$\beta = 1./(1 + \exp(1 * (map_{cloud} - 1)))$$

3. $$image_{\beta} = uint8(\beta. * double(image_{input}))$$

4. $$shadow_{\beta} = uint8(\alpha * (1 - \beta). * double(map_{shadow}))$$

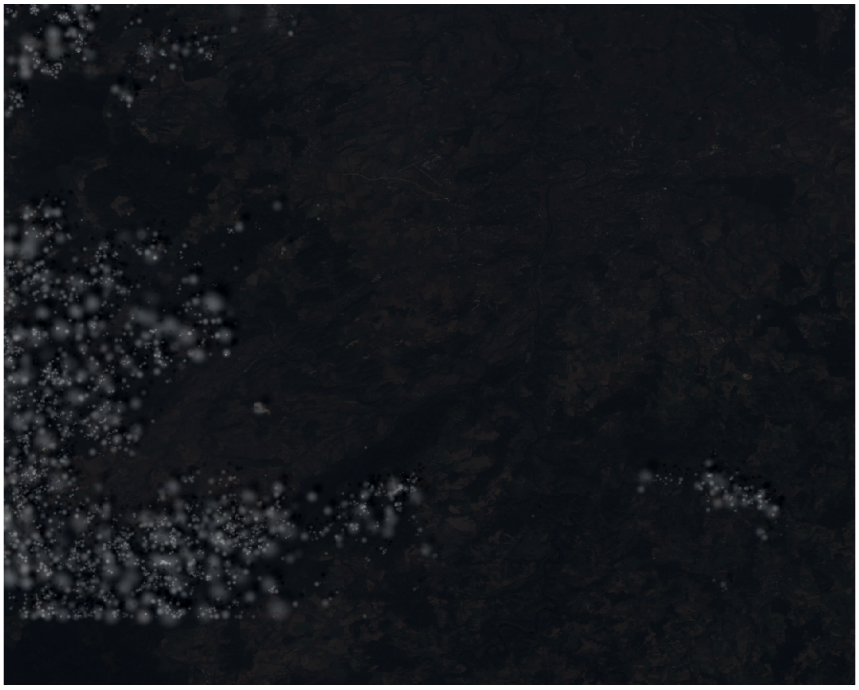
5. $$image_{output} = output\_map_{cloud} + image_{\beta} - shadow_{\beta},$$

Finally, we can match the histograms with the given reference image to enhance the visual quality. The differences of the applied procedure can be seen on the images 2.3:

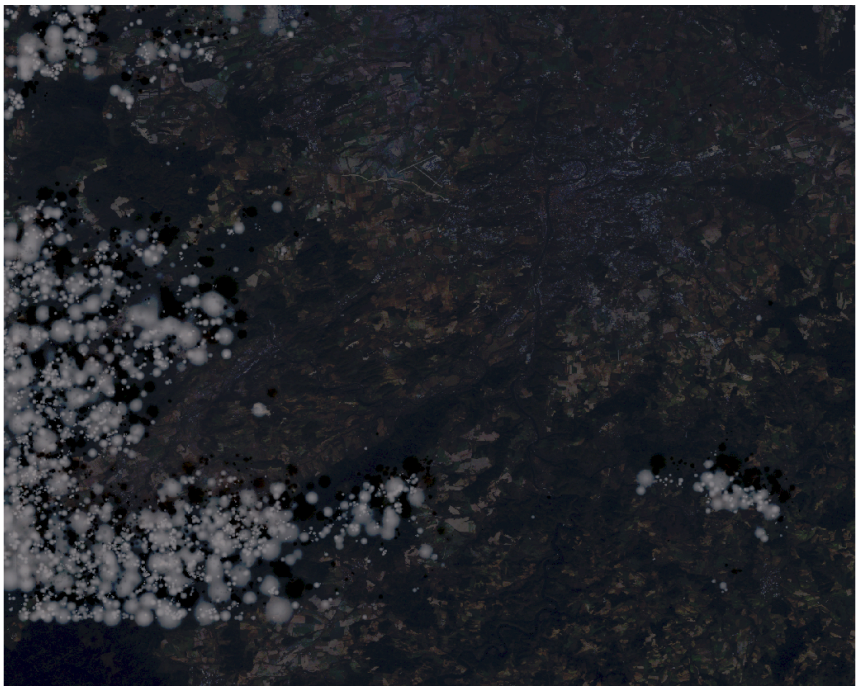
2.3 Evaluation

The main problem in the synthesizing option, as it is described, is that only "3D volume" clouds (cirrocumulus, altocumlus, stratocumulus, cumulonimbus, cumulus) are being modelled. The other types of clouds are hardly modelled by this procedure. The example of clouds modelling and residual images will follow: We have modelled an image with id 1_25 (lndcal.LT51910252006269MOR00_20060926_prg.tif) with maximally 3500 meta-balls, the initial probability is valid from 5 to 50 pixels radius, $k_1 = 0.001$, $k_2 = 0.001$. The stopping criterion of meta-balls is met prematurely with the maximal number of meta-balls. The progress of minimization the maximal value of the sum of the absolute differences can be found on a figure 2.6. We can find a residual image of a modelling process on a figure 2.4 and its histogram can be found on a figure 2.5. One can take a look at a figure 2.7 to see how many possible clouds could be modelled by our modelling algorithm after the stopping criterion is met.

There was an improvement for the thesis with the usage of the noisy image generation with the specified amplitude spectra [28]. One can create aesthetically pleasing low clouds this way and also generate their appropriate mask with labelling. This procedure can be extended even to the placing procedure, where to each meta-ball the noisy cloud image will be added. The



(a) : Synthesized image without the histogram matching



(b) : Syntehsized image with the histogram matching

Figure 2.3: The resulting images after synthesis

results of low clouds image with an added noise to each meta-ball and without can be seen on 2.8 and 2.9 respectively. However the resulting images were not added to the dataset creation, since the late discovery of the generating

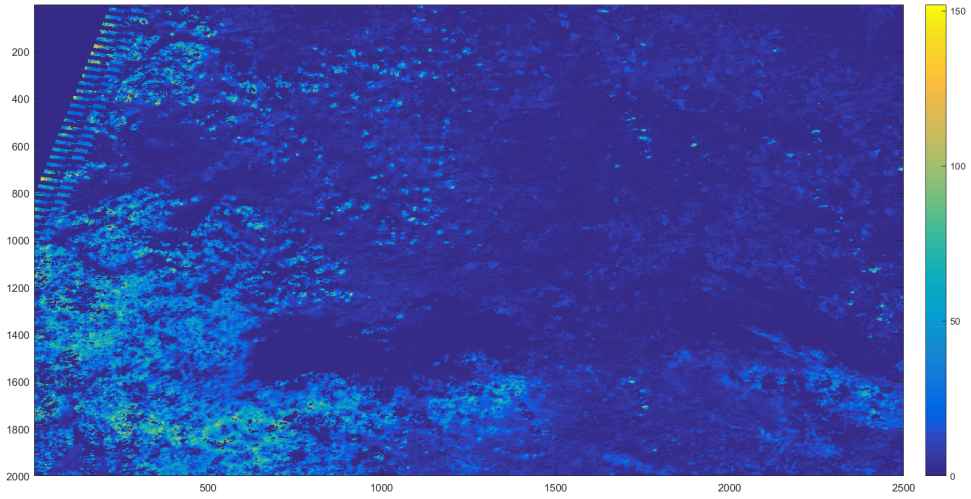


Figure 2.4: The residual image of the modelling process, where each pixel was averaged across channels and rounded down.

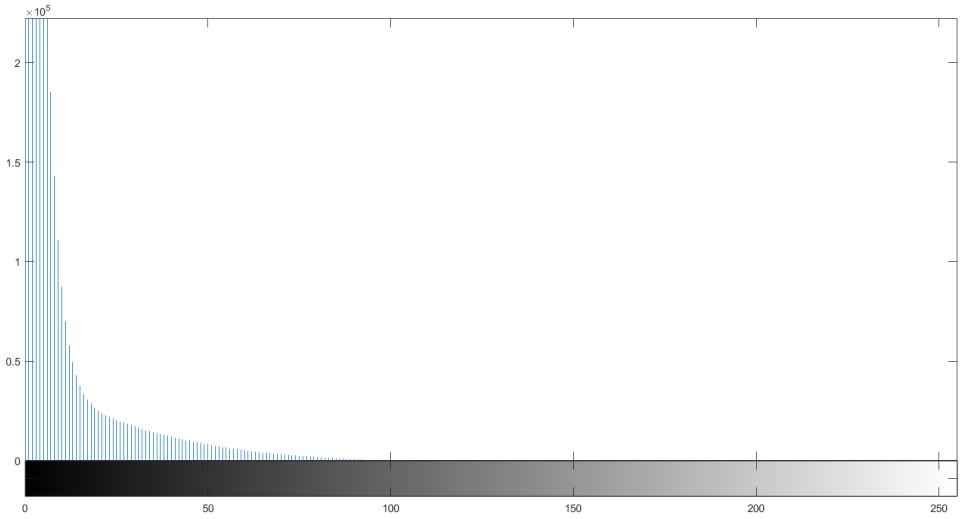


Figure 2.5: The histogram of the residual image of the modelling process.

code.

There is an option in the configuration file for choosing minimal and maximal cloud height and sun azimuth, which majorly influence where cloud shadows will be represented; e.g. one needs to raise the option "cloud height" to move all cloud shadows further in their direction. The main problematic part of combining cloud and cloud shadow was to simulate their overlapping parts. The first option was to prevent darkening of the background with a shadow, when a cloud is in the same position. This option leads to overly bright clouds, which were physically implausible, since the shadow was there from another cloud and the coefficients just needed to be carefully combined.

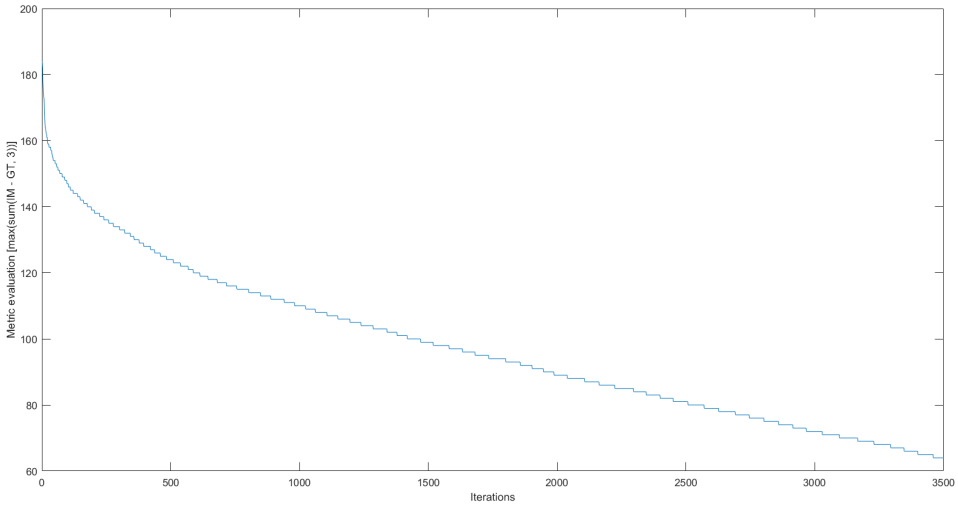


Figure 2.6: The error value curve as proportion to the number of iterations of the modelling procedure.

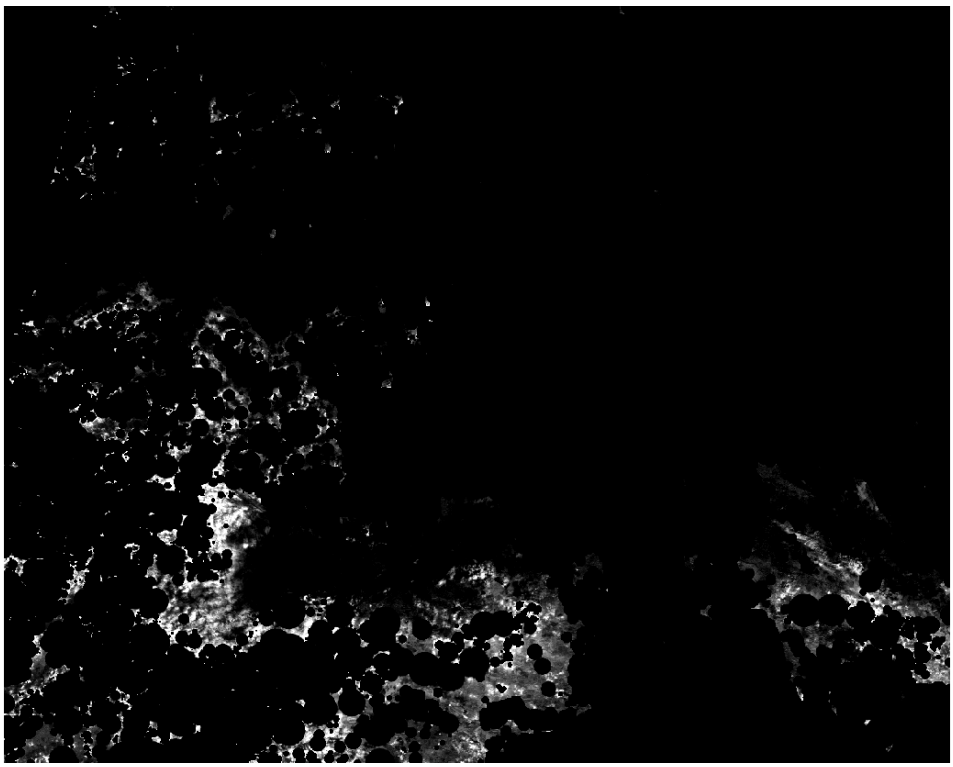


Figure 2.7: The residual of the modelling process inside our modelling algorithm. The image was visually enhanced. The white pixels represent possible location of a new meta-ball.

Therefore we employed a strategy for mixing these coefficients with a classical sigmoid to approximate the real life scenario, where the cloud shadow will dampen the overall brightness of a pixel. The main parameter for reducing

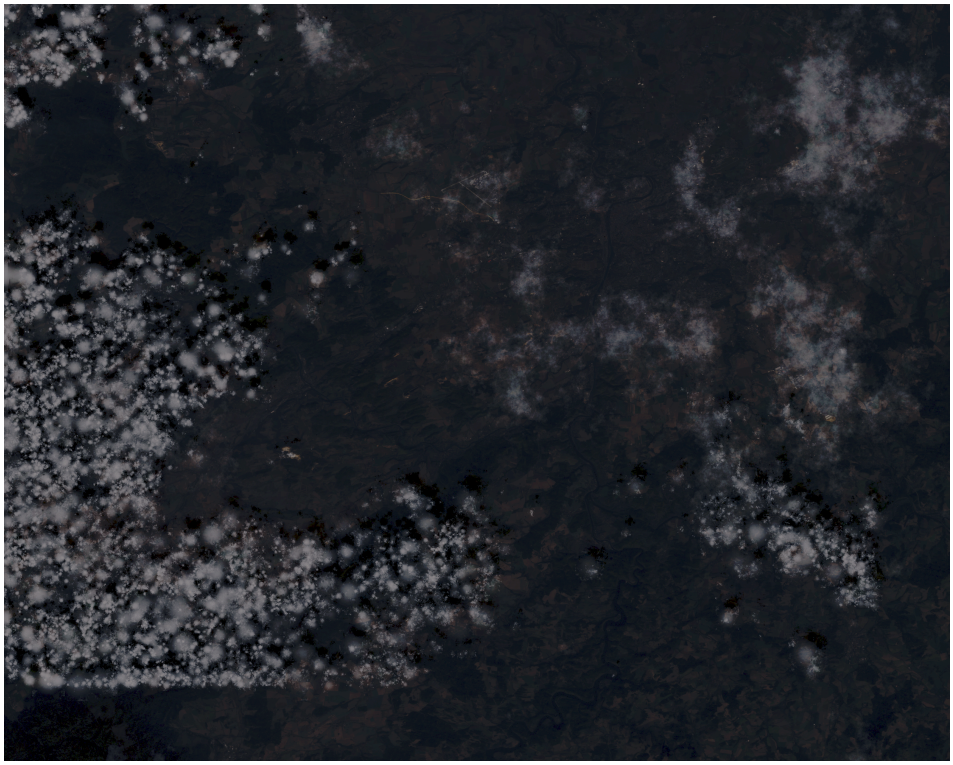


Figure 2.8: The noisy cloud image with the noise added to each meta-ball generated with the support of the code from [28].

the darkness of the cloud shadow is α, which determines how much of a reduction penalty will be applied to the brightness that will be reduced from the merged image as a shadow.

2.4 Dataset Creation

We have attempted to synthesize clouds from the old Landsat images, which were provided to us from the previous project with Gisat s.r.o. to properly classify our data. We have created our meta-balls cloud models according to a section 2.1. We chose 11 images that had clouds modelled by our approach. The choosing process consisted of finding images, which had visible cloud coverage, however not complete cloud coverage. We have chosen images, which had the most visually appealing character for modelling, where filenames and codes used in the program of these images can be found in a table 2.1.

We have chosen that our modelling technique will try to assign maximally 3500 individual meta-balls or metric (sum of absolute differences) attained

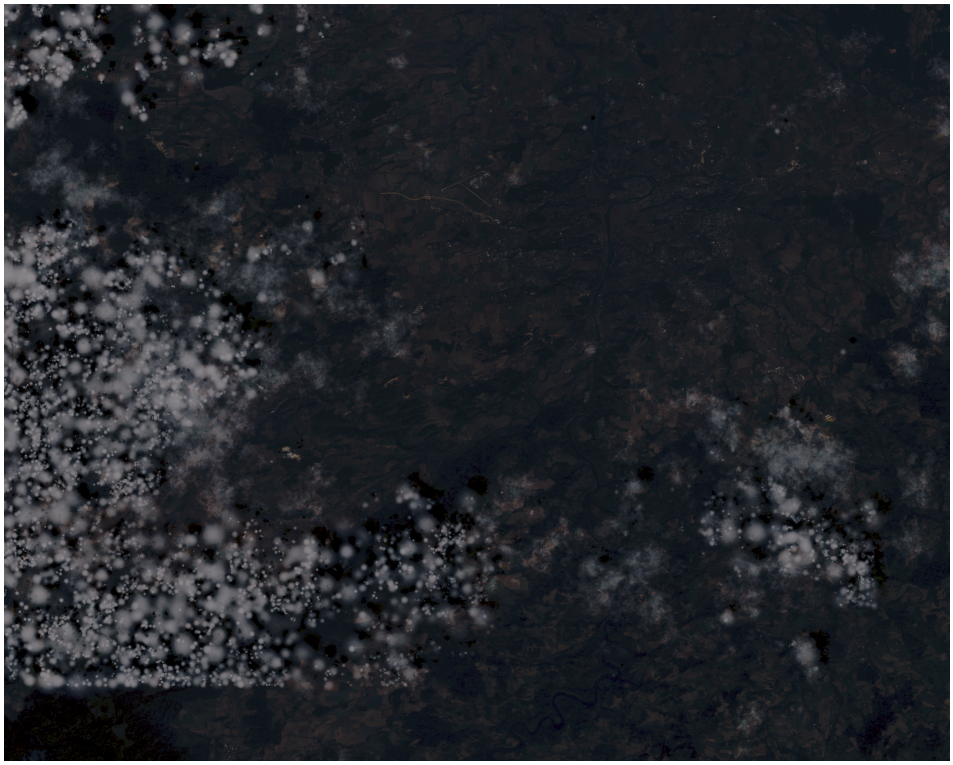


Figure 2.9: The noisy cloud image generated with the support of the code from [28].

values lower than 6. The visual approximation can be seen on 2.10.

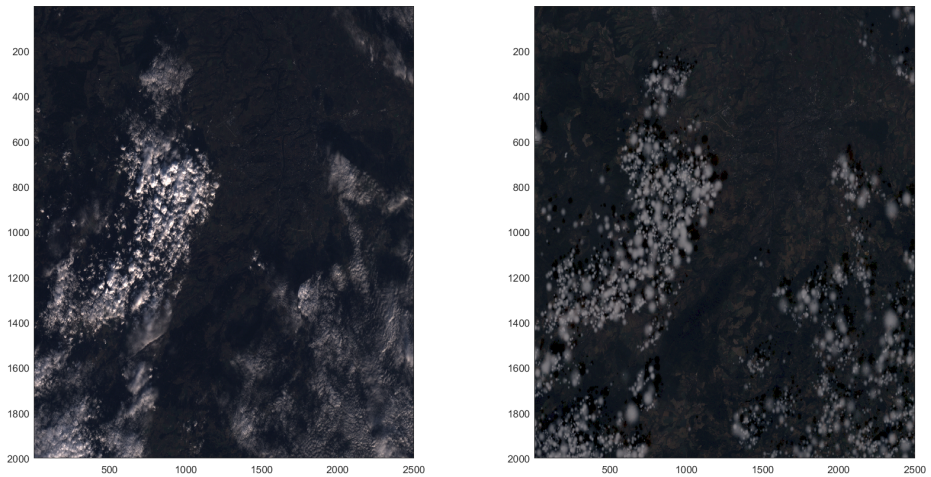


Figure 2.10: The visual differences of our approximation technique of meta-balls. On the left is an RGB representation of the multichannel satellite image and on the right is an RGB representation of our approximation.

We have generated 539 images with our synthesizing approach and their proper labelling. These images will be from now on referenced as training

Id	Image name
1_25	lndcal.LT51910252006269MOR00_20060926_prg.tif
1_26	lndcal.LT51910252006285MOR00_20061012_prg.tif
1_27	lndcal.LT51910252006301MOR00_20061028_prg.tif
2_30	lndcal.LT51920252007231MOR00_20070819_prg.tif
3_5_E	lndcal.LE71910252008283ASN00_20081009_prg.tif
3_6_E	lndcal.LE71910252008299ASN00_20081025_prg.tif
4_25	lndcal.LT51910252009309KIS00_20091105_prg.tif
4_26	lndcal.LT51910252009325KIS00_20091121_prg.tif
4_29	lndcal.LT51920252009204KIS01_20090723_prg.tif
4_36	lndcal.LT51920252009316KIS00_20091112_prg.tif
4_37	lndcal.LT51920252009332KIS00_20091128_prg.tif

Table 2.1: Used images in the cloud modelling

images. These training images were generated with the first synthesizing option, movement of generated meta-balls. Whereas we generated 110 images for testing purposes with the second option. Histograms were matched against the image 1_25 (lndcal.LT51910252006269MOR00_20060926_prg.tif) for all generated images to achieve similar brightness of clouds and similar contrast. The percentage of the split data on the testing to the whole amount is approximately 17%, therefore the following datasets will have the same proportion, due to the same windowing approach.

We incorporated a sliding window approach for testing and training images to generate appropriately sized input with a given stride. Then we used the compatibility of caffe and Lightning Memory-Mapped Database (LMBD) to create the training and testing databases for a fast access and therefore faster training. Before the windows were appended to the databases they were shuffled to prevent being stuck in local optima or recomputing non optimal gradient over and over. For instance the training database consisted of 614,460 images for a window with size 128x128 and stride 64 pixels and the testing database had 125,400 images.

We created a Gisat dataset for overall testing of our data synthesis consisting of all 39 new images from Gisat s.r.o., since the old images from Gisat s.r.o. were composed of only background label, cloud and shadow label merged together and "no observation" labels, which cannot be used when one wants to classify the shadow.

Chapter 3

Neural Network

Recently there has been a breakthrough in the field of machine learning as an artificial neural network, a theoretical discipline, had converted to a widely-used technology dominating the field. There are plenty of successful applications of this technology mainly in a pattern recognition. However the combination of the satellite imagery and neural network is still a growing field. Nevertheless there is a state of the art algorithm Fmask, which employs a thresholding strategy, on the other hand there is a room for improvement as there are visible mismatches in the shadow and cloud labellings. We wanted to utilize the abilities of neural networks as they perform well in noisy scenarios and as was shown recently in [22], [23], they can be used in remote sensing. Given the noisy ground truth we had, we have focused our research mainly on the combination of the satellite imagery and neural networks.

3.1 Realization

Even though we had the possibility to generate the synthesized data, we needed to decide which architecture of the CNN and deep learning generally, we need to employ to succeed in creating a good classifier. To recapitulate we used 4 categories, possible labels, of data: background, cloud shadow, cloud and no observation, where the categories evaluated to the numbers in range from 0 to 3 based on mentioning. We have concluded that there are 3 options based on the types of learning: Fully learn a network (also known as learning from scratch), to fine-tune a network or to use deep learned weights as features to the other classifier. The learning network from scratch means

learning all the parameters of the chosen network, whereas in fine-tuning one has learned weights and only wants to learn the last one to three layers of the network. The advantage of the fine-tuning over the learning from scratch is that the learning does not take a tremendous amount of time on an average network (approximately 40 - 80 layers), however the downfall is requirement of the learned weights. As a result of the time constraints we preferred to take the path of fine-tuning a network. SVM or other classifier or Conditional Random Fields (CRF) can be appended to the output of the learned network to create the third option.

The first attempt was trained on a personal laptop on NVIDIA 940MX graphical card, whereas the following attempts were trained with the contribution of the Center for Machine Perception (CMP) and its datagrid on graphical cards in CMP. There has been 4 modern graphical cards (NVIDIA Tesla K40C and three NVIDIA GTX Titan X), however there were shared resources for all the researchers, which slowed down the process of training neural networks, since there were sparse resources and time for learning them.

The learning of neural networks was performed in Caffe with Python bindings, because of the availability of pretrained weights and available models under Model Zoo section on their website.

The solver methods address the general optimization problem of loss minimization. For dataset D, the optimization objective is the average loss over all $|D|$ data instances throughout the dataset

$$L(W) = \frac{1}{|D|} \sum_{i}^{|D|} f_W(X^{(i)}) + \lambda r(W)$$

where $f_W(X^{(i)})$ is the loss on data instance $X^{(i)}$ and $r(W)$ is a regularization term with weight λ. $|D|$ can be very large, so in practice, in each solver iteration we use a stochastic approximation of this objective, drawing a mini-batch of $N << |D|$ instances:

$$L(W) \approx \frac{1}{N} \sum_{i}^{N} f_W(X^{(i)}) + \lambda r(W)$$

The model computes f_W in the forward pass and the gradient ∇f_W in the backward pass. The parameter update ΔW is formed by the solver from the error gradient ∇f_W, the regularization gradient $\nabla r(W)$, and other particulars to each method.

We had trained all networks with Stochastic gradient descent (SGD), which updates the weights W by a linear combination of the negative gradient $\nabla L(W)$ and the previous weight update V_t. The learning rate α is the weight of the negative gradient. The momentum μ is the weight of the previous update.

Formally, we have the following formulas to compute the update value V_{t+1} and the updated weights W_{t+1} at iteration $t+1$, given the previous weight update V_t and current weights W_t [2]

$$V_{t+1} = \mu\ V_t - \alpha \nabla L(W_t)$$
$$W_{t+1} = W_t + V_{t+1}$$

All networks in the text were trained with dataset consisting of synthesized images and of batchsize 25, if it is not said otherwise. The batchsize number comes from restrictions of the graphical card with the least memory capacity. We used the Rectified Linear Unit for all layers, also known as ReLU:

$$f(x) = max(0, x).$$

Our first attempt to create a classifier was a small neural network with an input image of size 5x5x3, where dimensions are as follows: height, width, channel. Primarily to this experiment we took all 6 channels as an input data, however after considering the fact, when in synthesizing the image the optimized densities were nearly of the same value, we reduced the input size to just 3 channels to standard RGB scheme of neural networks. We took only the first three channels of multichannel images, which in fact represented the blue, green and red. We created a synthesized dataset according to sliding window approach with a window size corresponding to input size and with stride 5 due to the enormous amount of resulting images. This network consists of two convolutional layers and final fully connected layer appended with classical softmax layer. We trained the network with parameters:

$$learning\_rate : \alpha = 1e-3$$

$$momentum : \mu = 0.9$$

$$batchsize = 2048$$

We were dropping the learning rate α by a step $\gamma = 0.1$ (which means multiplying the learning rate by γ) with the step of a length 10,000 iterations. We had stopped the learning at 45,000 iterations, since this network is fully trained. Unfortunately we did not have pretrained weights for such network.

The results were unsatisfactory even for testing synthesized dataset for which this network attained 0.875 overall accuracy. The evaluation of this network can be found in the following tables:

Evaluation metric	Score
Overall accuracy	0.875
Overall precision	0.703
Overall recall	0.572

Table 3.1: Overall Evaluation of 5x5 neural network

Label	Precision	Recall	F1-score	Support
Background	0.90	0.96	0.93	8479658
Shadow	0.51	0.16	0.24	460115
Cloud	0.69	0.60	0.64	1302275
Weighted Average / Total	0.86	0.88	0.86	10242048

Table 3.2: Evaluation of 5x5 neural network per label

Predicted label True label	Background	Shadow	Cloud
Background	0.792	0.004	0.03
Shadow	0.035	0.007	0.003
Cloud	0.049	0.002	0.077

Table 3.3: Confusion matrix for 5x5 neural network

The second attempt was to create a network similar to the ImageNet. We used a CaffeNet, which is a minor variant on a popular architecture AlexNet [29] by Alex Krizhevsky et al. We wanted to incorporate the low convolution filters trained on ImageNet into our detection, we initialized our process with weights pretrained on the ImageNet dataset, which should hugely decrease the learning time and setup a good accuracy. We let the same sized input, 128x128, because we thought that the previous attempt of 5x5 was insufficient to properly classify, and changed the number of outputs to correspond to our labels. This network had 5 convolutional layers with 3 pooling layers and 3 fully connected layers. We let the channels and the learning parameters be same as in the previous attempt. We needed to update the window size for dataset creation and create a new datasets for learning with stride 64. However we tried to fine-tune the network based on publicly available ImageNet weights with different number of iterations. The overall results after finetuning for 25,000 iterations can be found in table 3.4. Even though the accuracy is fairly high there occurs a complete omission of shadow label as it can be seen in the following tables 3.5, 3.6. We discarded this architecture base on these facts.

The main part of our research was focused on SegNet architecture [30],[31],[32], which incorporated a famous encoder-decoder scheme. This architecture as it is stated in the articles can be seen on figure 3.1. We focused on the input of size 128x128 with a classical three input dimensions of an image and

Evaluation metric	Score
Overall accuracy	0.827
Overall precision	0.439
Overall recall	0.422

Table 3.4: Overall Evaluation of 128x128 neural network

Label	Precision	Recall	F1-score	Support
Background	0.87	0.97	0.92	101176
Shadow	0.00	0.00	0.00	6197
Cloud	0.45	0.29	0.36	17811
Weighted Average / Total	0.76	0.83	0.79	125184

Table 3.5: Evaluation of 128x128 neural network per label

Predicted label True label	Background	Shadow	Cloud
Background	0.786	0	0.022
Shadow	0.020	0	0.029
Cloud	0.101	0	0.042

Table 3.6: Confusion matrix for 128x128 neural network

output is also an image of size 128x128 with a class assignment. We have inspired ourselves with [33], where they achieve outstanding pixel-wise classification results with SegNet architecture. On their GitHub page there are final weights of their work, however there were major differences in ours and theirs datasets. Since their dataset contained a digital surface model heights and a ground sampling distance of 5 cm, whereas ours sampling distance in Landsat imagery is 30 m, in addition they were classifying into an urban areas. Despite that we tried to use their weights as an initial configuration for learning instead of a random initialization or the ImageNet weights. We had created a training and a testing dataset consisting of the synthesized images of size 128x128 with a sliding window approach with stride 64. We took only the first three channels. We also created a separate dataset consisting of only original not synthesized images from Gisat s.r.o. The training dataset was used for the learning for 60,000 iterations with the same learning settings as in the previous examples, except:

$$learning\_rate : \alpha = 1e - 4$$

$$step\_size : 1000$$

For an easier orientation we will name this combination of neural network with the learned weights a $SegNet60$. The basic overall evaluation of $SegNet60$ can be found in a table 3.7. The Confusion matrix and evaluation per label can be seen in tables 3.9, 3.8.

However, the resulting accuracy did not rise with the increasing number

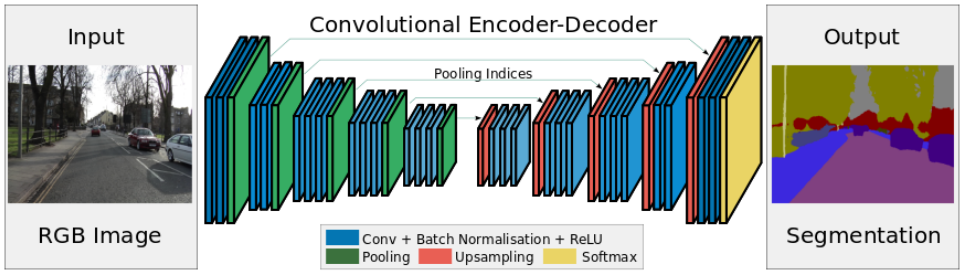


Figure 3.1: SegNet architecture [34]

Evaluation metric	Score
Overall accuracy	0.9597
Overall precision	0.9079
Overall recall	0.8517

Table 3.7: Overall Evaluation of SegNet60 weights

Label	Precision	Recall	F1-score	Support
Background	0.97	0.99	0.98	1660270669
Shadow	0.81	0.67	0.74	109595183
Cloud	0.94	0.89	0.92	284687748
Weighted Average / Total	0.96	0.96	0.96	2054553600

Table 3.8: Evaluation of SegNet60 weights

Predicted label True label	Background	Shadow	Cloud
Background	0.8	0.004	0.004
Shadow	0.014	0.036	0.004
Cloud	0.010	0.005	0.124

Table 3.9: Confusion matrix for SegNet60 weights

of iterations. We have introduced a well-known improvement Hard Negative Mining in the dataset, where we classified the images based on a random index in the dataset and if the accuracy was lower than the given threshold we added this image to a new dataset consisting of these hard examples. However, we added the lucky option to prevent over-saturation from hard examples. The lucky option added the image to the new dataset, even though it was classified as "good", thus its accuracy was more than given threshold, then we draw a random number and based on thresholding the random number we decided whether the image will be rejected. We have repeated this procedure of Hard Negative Mining and learned our network with it. The accuracy has risen for a small number of iterations from overall accuracy 0.9597 to 0.9606. Nevertheless, the overall accuracy for Gisat dataset decreased from nearly 0.8 to almost 0.7. These learned weights will be named as a $SegNetHNM$ (as the Hard Negative Mining) in the following work. The evaluation of the newly trained network can be found in the following tables 3.10, 3.11, 3.12

Evaluation metric	Score
Overall accuracy	0.9606
Overall precision	0.91
Overall recall	0.8561

Table 3.10: Overall Evaluation of SegNetHNM weights

Label	Precision	Recall	F1-score	Support
Background	0.97	0.99	0.98	1660270669
Shadow	0.81	0.69	0.74	109595183
Cloud	0.95	0.89	0.92	284687748
Weighted Average / Total	0.96	0.96	0.96	2054553600

Table 3.11: Evaluation of SegNetHNM weights per label

Predicted label True label	Background	Shadow	Cloud
Background	0.8	0.004	0.004
Shadow	0.014	0.037	0.003
Cloud	0.010	0.005	0.123

Table 3.12: Confusion matrix for SegNetHNM weights

We separated Gisat dataset into two parts consisting of 90% training dataset and 10% testing dataset. We constructed a mixed dataset consisting of the dataset of hard examples and gisat training dataset and tried to further fine-tune our network. Thus we created initialization for fine-tuning the network with Gisat dataset. We added 10 low clouds images created with an algorithm from [28]. The images were sampled with sliding window approach to the newly constructed dataset in attempt to model low clouds in training dataset. The newly learned weights with this approach will be called $SegNetHardLow$. The evaluations of $SegNetHardLow$ on Gisat dataset can be found in the tables 3.13, 3.14, 3.15, 3.16.

Although we had incorporated the "no observation" label into classification we rejected the possibility of classification of such label, since there were no training examples on which learning was possible. We fixed classification of "no observation" class. After classification, we assigned this label based on ground truth, since the class was known apriori from the sensor. The overall evaluation of all learned weights can be found in table 3.17

Evaluation metric	Score
Overall accuracy	0.943
Overall precision	0.7057
Overall recall	0.7383

Table 3.13: Overall Evaluation of SegNetHardLow weights on Gisat dataset

Label	Precision	Recall	F1-score	Support
Background	1.00	0.94	0.97	1188510210
Shadow	0.08	0.01	0.03	22417831
Cloud	0.74	0.89	0.85	253861940
No observation	1.00	1.00	1.00	195318819
Weighted Average / Total	0.95	0.94	0.94	1660108800

Table 3.14: Evaluation of SegNetHardLow weights per label on Gisat dataset

Predicted label True label	Background	Shadow	Cloud
Background	0.6724	0.0022	0.0414
Shadow	0.0012	0.0002	0.0121
Cloud	0.0001	0	0.1528

Table 3.15: Confusion matrix for SegNetHardLow weights on Gisat dataset

Label	Score
Background	0.7313
Shadow	0.0212
Cloud	0.791

Table 3.16: Intersection over Union of SegNetHardLow weights on Gisat dataset

We have tried to introduce a new variable δ into the classification process. This variable aimed at creating a possibility of classification alteration. We have defined γ as a quotient of a second highest number and a highest number of the neural network softmax output of a single pixel. We have thresholded γ by δ. If $\gamma < \delta$, the label with a maximum value stays assigned. Otherwise the background label is assigned, therefore with a choice of δ different classifications can be made. Such exemplary results of different evaluations can be seen on a figure 3.2.

Evaluation metric	SegNet60	SegNetHNM	SegNetHardLow
Overall accuracy	0.791	0.717	0.943
Overall precision	0.669	0.653	0.706
Overall recall	0.676	0.658	0.738

Table 3.17: Overall Evaluation of learned SegNet weights on Gisat testing dataset

A few representing figures of the classified Gisat images are 3.3, 3.4. The labelled images were evaluated with our neural network with the input size of 128x128. Thus, we needed to implement a sliding window approach, so we could classify these images. The sliding window approach was implemented in such way that stride was 64 pixels, therefore the windows will overlap. The pixels of overlapping windows will add a vote to a total labelling to exploit the overlapping window concept.

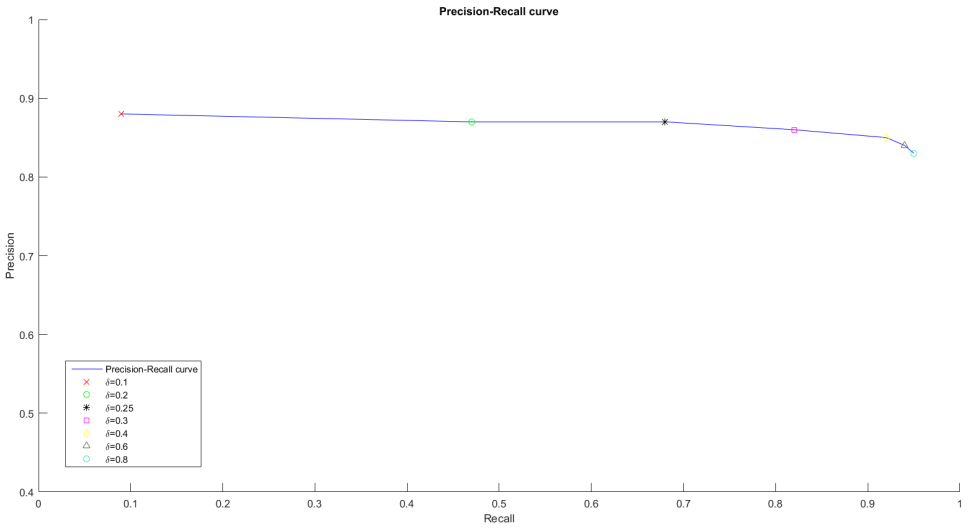
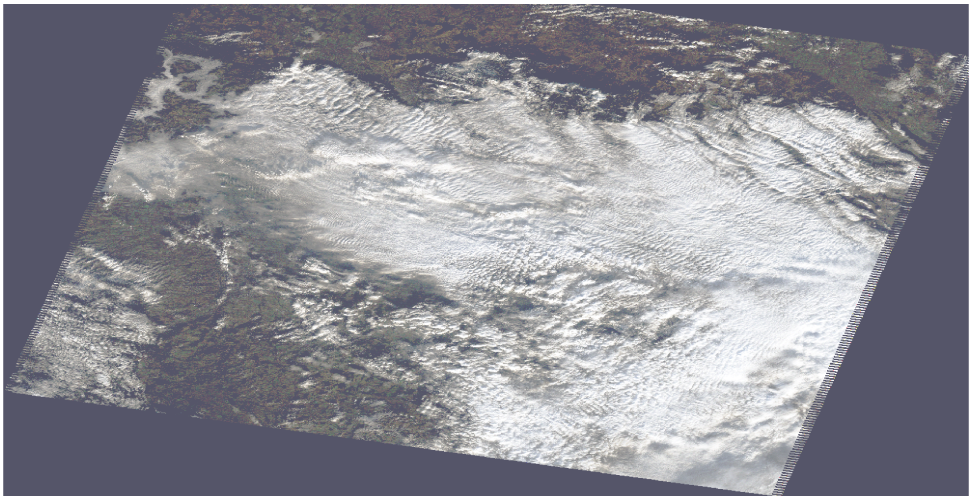


Figure 3.2: Precision-Recall curve of cloud label for different values of δ

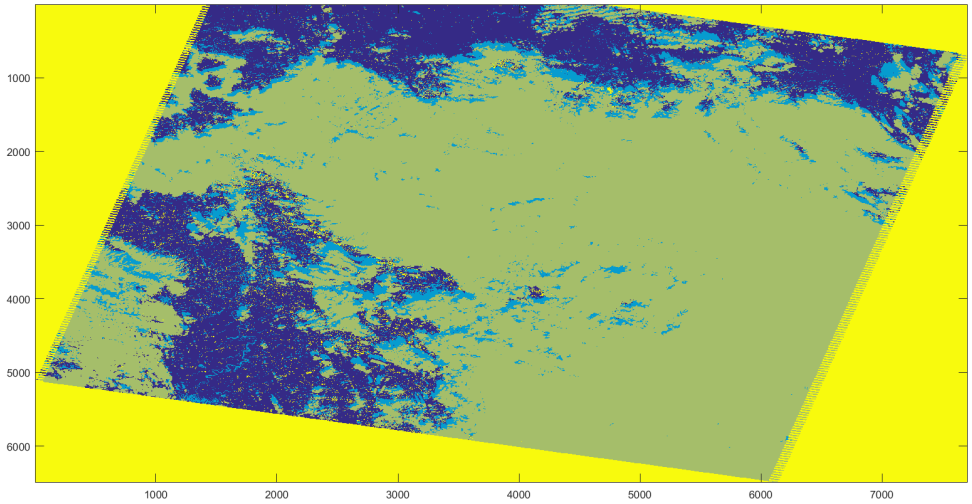
The main problem of classification are the cloud shadows as it can be seen on all previous evaluation tables and figures 3.3, 3.4. The classification of cloud shadow is poor due to a low distinguishability of that label. The possible improvement could be to copy the Flood fill algorithm based on sensor's metadata from the Fmask algorithm.

We have tried to smooth out our predictions and make the final improvements with a Graph Cut algorithm, where pairwise cost is the Euclidean distance through each band and unary cost represents the output of neural network. We tried to enhance the cloud shadow classification by taking into account metadata of the sensor. We projected a cloud label in a given direction and based on a maximal thrown shadow giving us the probable shadow pixels of the cloud pixel. Then, we created an edge from the cloud pixel to the probable shadow pixel when the probable shadow pixel was not cloud in the neural network output. We used classical software GCOptimazation [35], [36], [37], [38] to help us with this problem. The examples of such smoothing can be seen in the figures 3.5, 3.6. The overall accuracy has risen from 0.8218 to 0.8228 in the first figure, whereas it has decreased from 0.8752 to 0.8718 in the second figure.

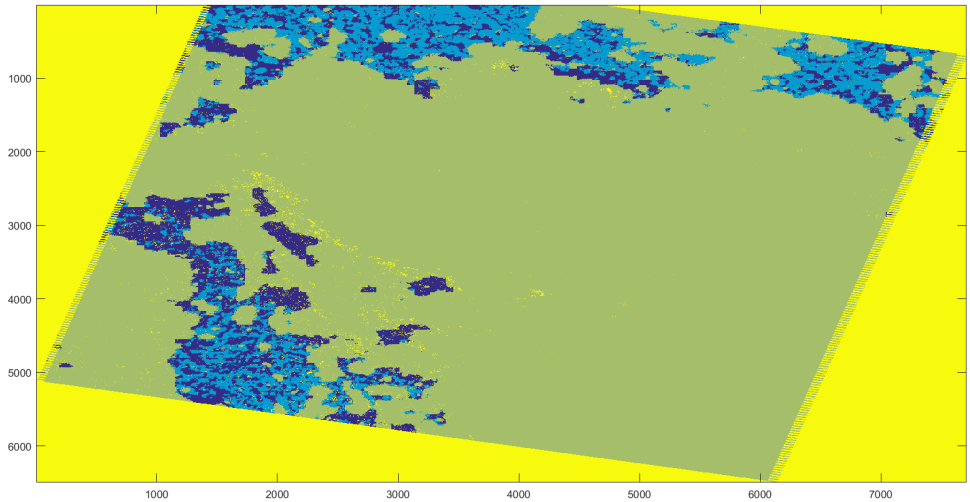
As a last experiment, we have tested generalization of our neural network on images made by a different sensor. We have been given 10 images from satellite Sentinel-2 by Gisat s.r.o. We have adapted classes to be equivalent with ours. The images have 9 channels, where some of these channels can be chosen to have an approximately same wavelength range. The spatial resolution of these images is 20 meters. We preprocessed images with the same approach as with Landsat images. We achieved mean accuracy of 0.6460. One of the average classifications can be found in the figure 3.7.



(a) : Original fourth testing image from Gisat s.r.o. displayed in RGB with histogram equalization

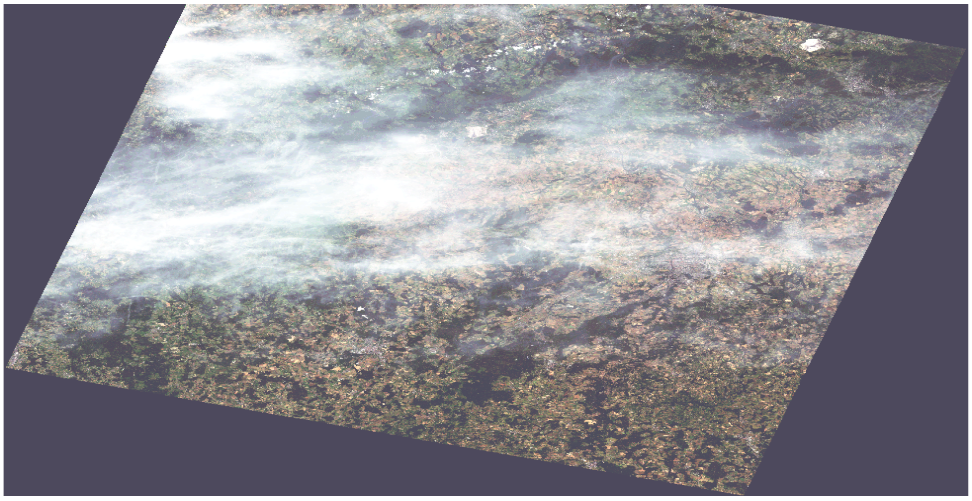


(b) : Provided labelling from Gisat s.r.o. reduced to same labels

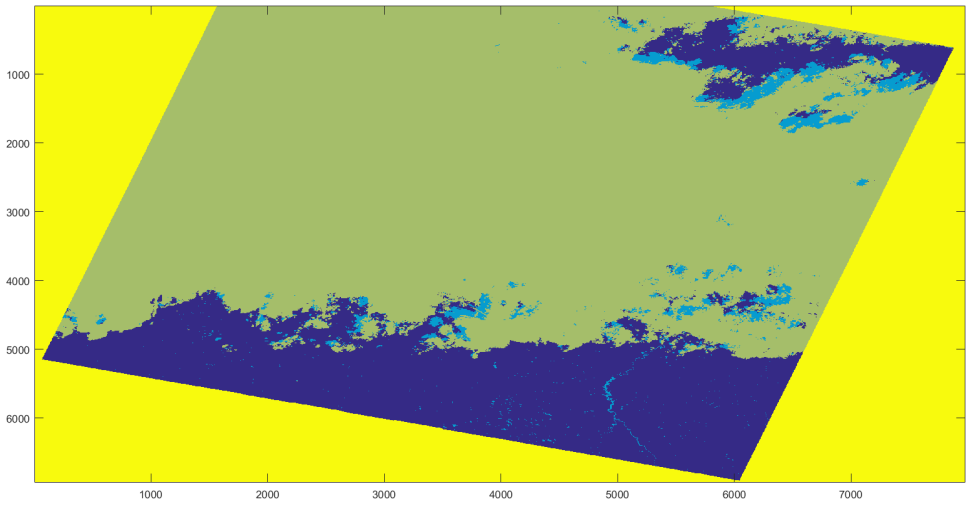


(c) : Labelled image by our procedure

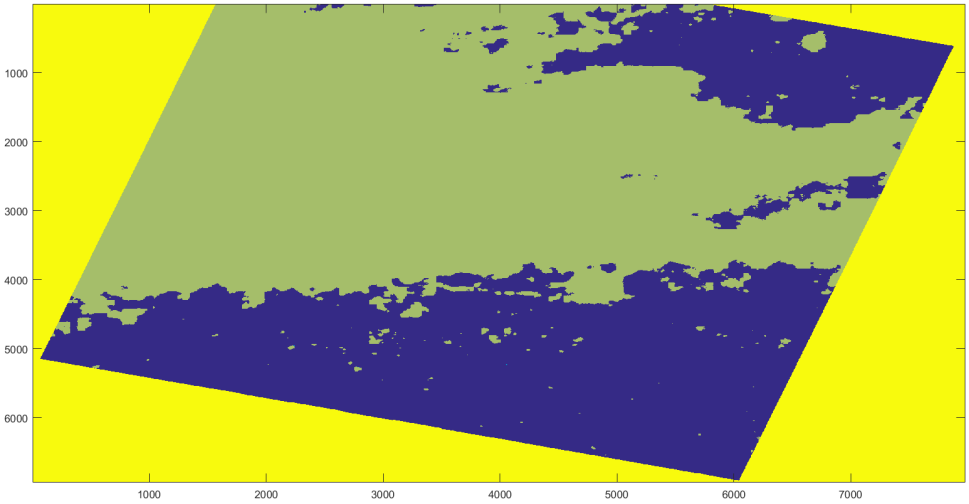
Figure 3.3: The differences in semantic labeling for 4th testing image



(a) : Original seventh testing image from Gisat s.r.o. displayed in RGB with histogram equalization

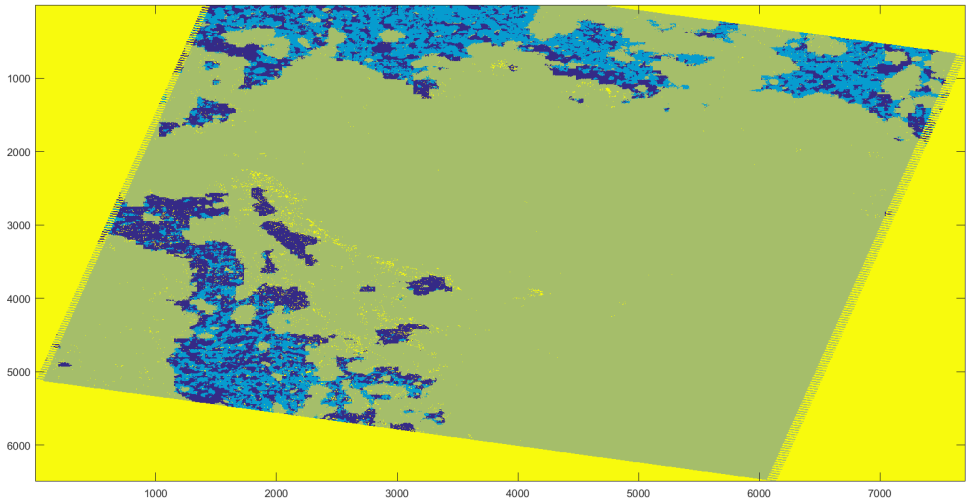


(b) : Provided labelling from Gisat s.r.o. reduced to same labels

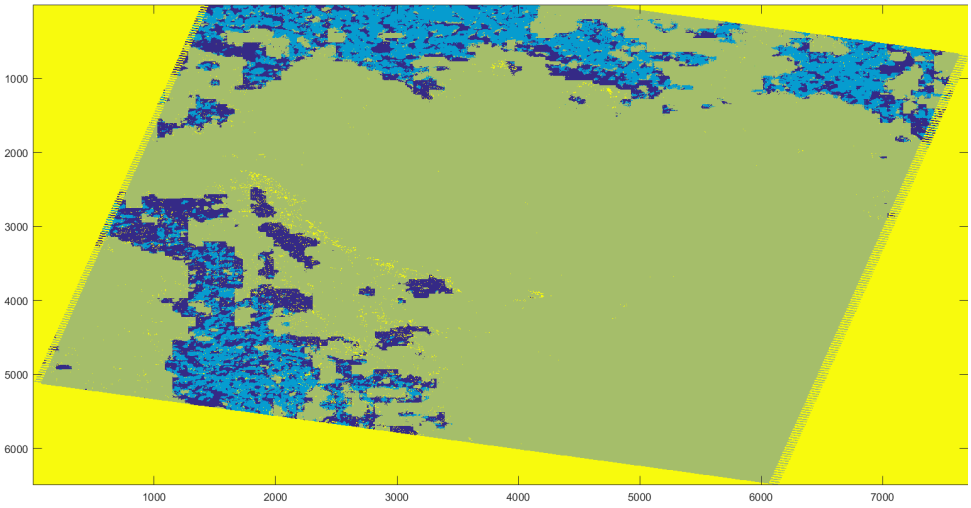


(c) : Labelled image by our procedure

Figure 3.4: The differences in semantic labeling for 7th testing image

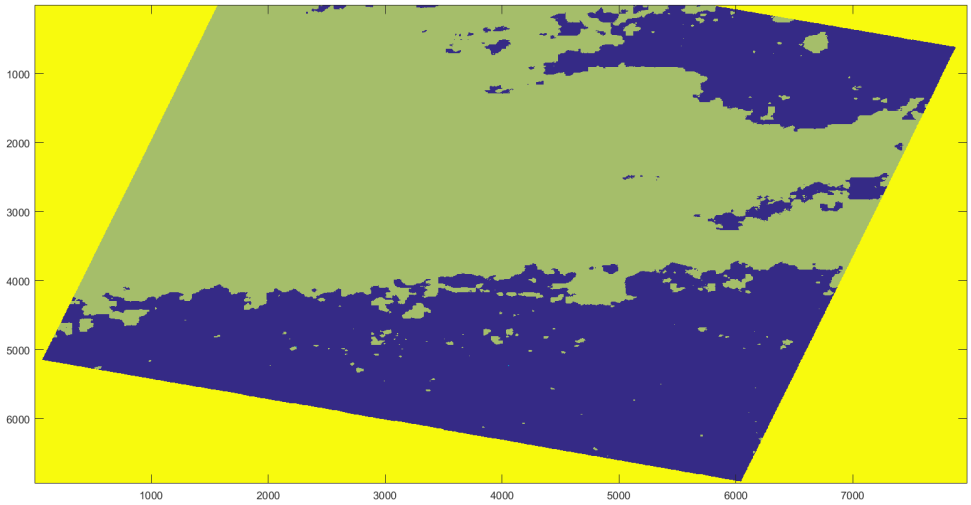


(a) : Labelled image by our procedure

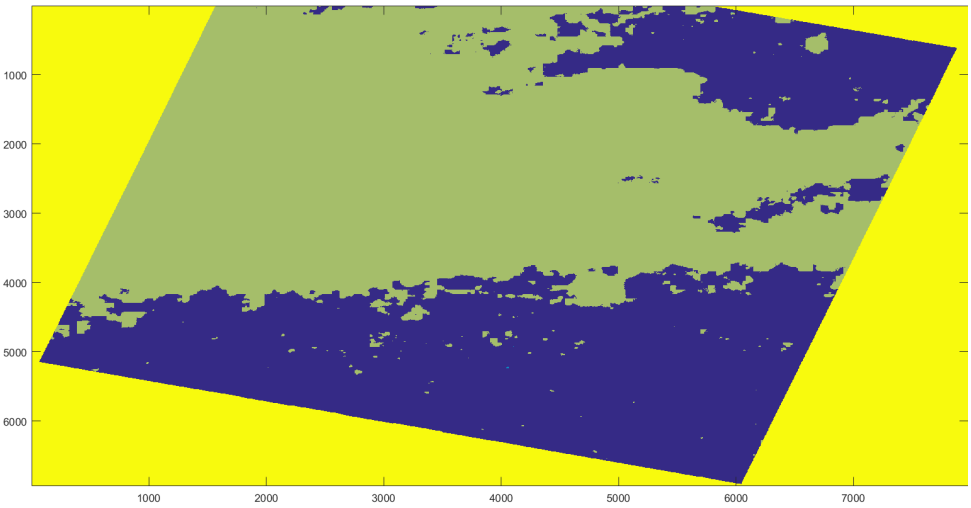


(b) : Labelled image by Graph Cut postprocessing

Figure 3.5: The differences in usage of Graph Cut postprocessing for 4th testing image

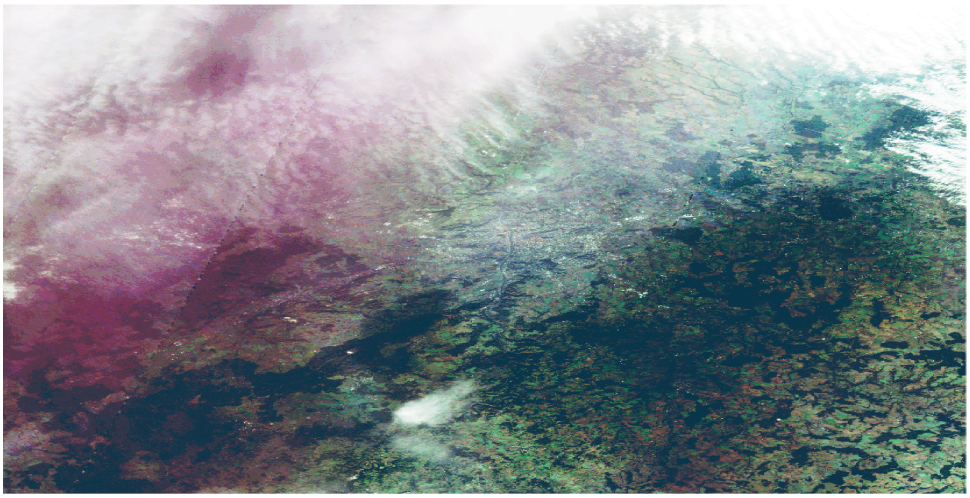


(a) : Labelled image by our procedure

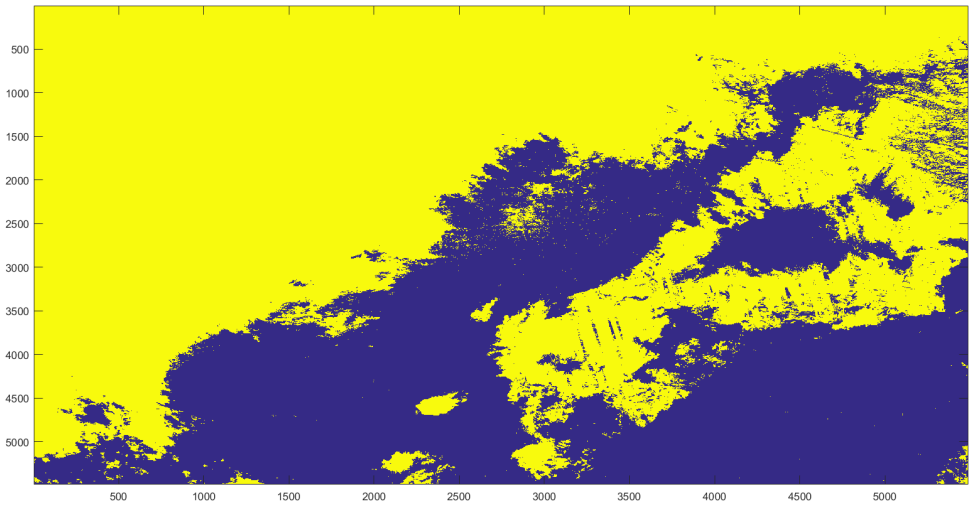


(b) : Labelled image by Graph Cut postprocessing

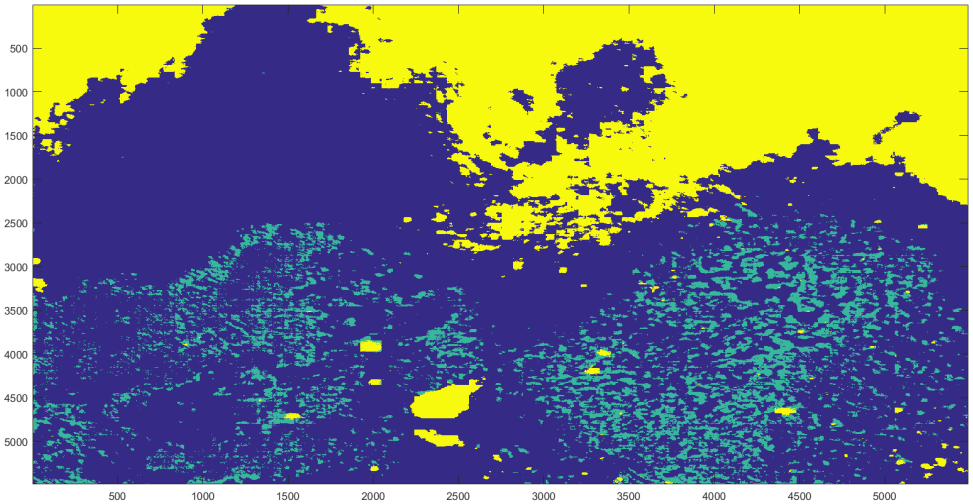
Figure 3.6: The differences in usage of Graph Cut postprocessing for 7th testing image



(a) : Original third testing image from Sentinel-2 from Gisat s.r.o. displayed in RGB with histogram equalization



(b) : Provided labelling from Gisat s.r.o. reduced to same classes



(c) : Labelled image by our procedure

Figure 3.7: The differences in semantic labeling for 3rd image from Sentinel-2

Chapter 4

Conclusion

We have created a modelling and synthesizing pipeline for the cloud based satellite imagery, which enables generating an arbitrary amount of new images and their appropriate masks. We have graphically demonstrated the power of this procedure. Thus we have proposed a synthesizing procedure. This procedure can be used mainly in two cases. Firstly, there is only a notion of data, but no real data are given. Secondly, there is a small amount of data and we need to raise the amount of the obtained data.

We have employed a fine-tuning approach of learning neural networks due to the time and computation constraints. We have tried out a few basic architectures ranging from a basic convolutional net with a few parameters to the SegNet Coder-Decoder architecture. Nevertheless, we focused our research mainly on SegNet architecture, due to its outstanding pixel-wise classification ability. We developed the datasets based on the synthesized data and learned the networks. We learned our networks on the training dataset and evaluated them on the testing dataset. The learned weights for SegNet architecture reached 95.97% overall accuracy (overall precision 90.79% and overall recall 85.17%) on the testing dataset. However, the continuing iterations did not raise the overall accuracy. Thus, we have incorporated a Hard Negative Mining, which have improved the overall accuracy to 96.06% (overall precision 91% and overall recall 85.61%) after a few iterations. We have added the Gisat training images to the training dataset with generated low cloud images. After another few iterations, we have improved the overall accuracy on testing Gisat dataset, however we have lost some accuracy on the generated dataset.

We introduced a new parameter δ for thresholding the neural network output. We have plotted the interesting choices of this parameter on ROC curve. We demonstrated the neural network ability in the evaluation tables and on the resulting figures. We have implemented the Graph Cut algorithm into our classification with the intention to smooth out the predictions, even though the smoothed labels raised the overall accuracy by maximally 0.05.

However we have not solved the problem of cloud shadow classification, which seemed to be basically undetectable on Gisat dataset under our current approach. The possible improvement could be to replicate Fmask's Flood fill algorithm to classify shadows properly.

The overall speed of learning could be potentially fastened by using other deep learning framework like TensorFlow by Google or newly announced Caffe2 by joint collaboration of NVIDIA and Facebook.

Bibliography

[1] Zhe Zhu & Curtis E. Woodcock.
"Object-based cloud and cloud shadow detection in Landsat imagery".
In: *Remote Sensing of Environment* 118 (Mar. 2012), pp. 83–94.

[2] Yangqing Jia & Evan Shelhamer & Jeff Donahue & Sergey Karayev &
Jonathan Long & Ross Girshick &
Sergio Guadarrama & Trevor Darrell.
"Caffe: Convolutional Architecture for Fast Feature Embedding".
In: *arXiv preprint arXiv:1408.5093* (2014).

[3] GISGeography.
What is Remote sensing? The Definitive Guide to Earth Observation.
2017. URL: http://gisgeography.com/remote-sensing-earth-observation-guide/ (visited on 03/16/2017).

[4] Canadian Space Agency. *RADARSAT-1.* 2014.
URL: http://www.asc-csa.gc.ca/eng/satellites/radarsat1/
(visited on 03/16/2017).

[5] Canadian Space Agency. *RADARSAT-2.* 2015.
URL: http://www.asc-csa.gc.ca/eng/satellites/radarsat2/
(visited on 03/16/2017).

[6] U.S. Department of the Interior & U.S. Geological Survey. *Landsat.*
2016. URL: https://landsat.usgs.gov/ (visited on 03/18/2017).

[7] U.S. Department of the Interior & U.S. Geological Survey.
Landsat Missions Timeline. 2016.
URL: https://landsat.usgs.gov/landsat-missions-timeline
(visited on 03/23/2017).

[8] Czech Technical University & Gisat s.r.o. "Urban Dynamic Processor".
In: (2015).

[9] Olga Russakovsky et al.
"ImageNet Large Scale Visual Recognition Challenge".
In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[10] U.S. Department of the Interior & U.S. Geological Survey.
USGS Global Visualization Viewer. 2017. URL:
http://glovis.usgs.gov/index.shtml (visited on 01/20/2017).

[11] U.S. Department of the Interior & U.S. Geological Survey.
GloVis Next. 2017.
URL: http://glovis.usgs.gov/next (visited on 01/20/2017).

[12] U.S. Department of the Interior & U.S. Geological Survey.
Earth Explorer. 2017.
URL: https://earthexplorer.usgs.gov (visited on 01/20/2017).

[13] K. He & J. Sun & X. Tang.
"Single Image Haze Removal Using Dark Channel Prior".
In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2011), pp. 2341–2353.

[14] Zhe Zhu & Curtis E. Woodcock. "Automated cloud, cloud shadow, and snow detection in multitemporal Landsat data: An algorithm designed specifically for monitoring land cover change".
In: *Remote Sensing of Environment* 152 (2014), pp. 217–234.
ISSN: 0034-4257.
DOI: http://dx.doi.org/10.1016/j.rse.2014.06.012.
URL: http://www.sciencedirect.com/science/article/pii/S0034425714002259.

[15] Zhe Zhu & Shixiong Wang & Curtis E. Woodcock. "Improvement and expansion of the Fmask algorithm: cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images".
In: *Remote Sensing of Environment* 159 (2015), pp. 269–277.
ISSN: 0034-4257.
DOI: http://dx.doi.org/10.1016/j.rse.2014.12.014.
URL: http://www.sciencedirect.com/science/article/pii/S0034425714005069.

[16] NASA Earth Observatory.
Measuring Vegetation (NDVI & EVI) : Feature Articles. 2011.
URL: http://earthobservatory.nasa.gov/Features/MeasuringVegetation
(visited on 02/22/2017).

[17] Q. Yuan & G. Yang X. Li & H. Shen & L. Zhang & H. Zhang.
"Recovering quantitative remote sensing products contaminated by thick clouds and shadows using multitemporal dictionary learning".
In: *IEEE Transactions on Geoscience and Remote Sensing* 52.11 (Nov. 2014).

[18] J.Wang & P. A. Olsen & A. R. Conn & A. C. Lozano. "Removing Clouds and Recovering Ground Observations in Satellite Image Sequences via Temporally Contiguous Robust Matrix Completion". In: (2016).

[19] Ben V. Hollingsworth & Liqiang Chen & Stephen E. Reichenbach & Richard R. Irish.
"Automated cloud cover assessment for Landsat TM images".
In: *Proceedings of SPIE, Imaging Spectrometry II* 2819 (Nov. 1996), pp. 170–181.

[20] Yoshinori Dobashi & Yusuke Shinzo & Tsuyoshi Yamamoto.
"Modeling of Clouds from a Single Photograph".
In: *Computer Graphics Forum* 29.7 (2010), pp. 2083–2090.
ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2010.01795.x.
URL: http://dx.doi.org/10.1111/j.1467-8659.2010.01795.x.

[21] Brendan Mccane and Peter Stephenson.
An Atmospheric Cloud Model for Image Synthesis and Deterministic and Nondeterministic Animation Abstract.

[22] Marco Castelluccio & Giovanni Poggi &
Carlo Sansone & Luisa Verdoliva. "Land Use Classification in Remote Sensing Images by Convolutional Neural Networks".
In: *Computing Research Repository (CoRR)* abs/1508.00092 (2015).
URL: http://arxiv.org/abs/1508.00092.

[23] Keiller Nogueira & Otávio Augusto Bizetto Penatti &
Jefersson Alex dos Santos. "Towards Better Exploiting Convolutional Neural Networks for Remote Sensing Scene Classification".
In: *Computing Research Repository (CoRR)* abs/1602.01517 (2016).
URL: http://arxiv.org/abs/1602.01517.

[24] C. Gonzalo-Martin et al. "Deep learning for superpixel-based classification of remote sensing images". In: Sept. 2016.
URL: http://proceedings.utwente.nl/401/.

[25] US Department of Commerce & NOAA &
Earth System Research Laboratory. *Zenith angle*. 2016. URL:
https://www.esrl.noaa.gov/gmd/grad/solcalc/azelzen.gif
(visited on 03/29/2017).

[26] Yoshinori Dobashi & Tomoyuki Nishita &
Hideo Yamashita & Tsuyoshi Okita.
"Modeling of clouds from satellite images using metaballs".
In: (Oct. 1998), pp. 53–60, 224. DOI: 10.1109/PCCGA.1998.731998.

[27] Geoff Wyvill & Andrew Trotman. "Ray-Tracing Soft Objects". In: *CG International '90: Computer Graphics Around the World*.
Ed. by Tat-Seng Chua & Tosiyasu L. Kunii.
Tokyo: Springer Japan, 1990, pp. 469–476. ISBN: 978-4-431-68123-6.
DOI: 10.1007/978-4-431-68123-6_27.
URL: http://dx.doi.org/10.1007/978-4-431-68123-6_27.

[28] Peter Kovesi. *Peter Kovesi*.
URL: http://www.peterkovesi.com/ (visited on 04/20/2017).

[29] Alex Krizhevsky & Ilya Sutskever & Geoffrey E. Hinton.
"ImageNet Classification with Deep Convolutional Neural Networks".
In: *Advances in Neural Information Processing Systems 25*.
Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105.
URL:
http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[30] Alex Kendall & Vijay Badrinarayanan & Roberto Cipolla.
"Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding".
In: *Computing Research Repository (CoRR)* abs/1511.02680 (2015).
URL: http://arxiv.org/abs/1511.02680.

[31] Vijay Badrinarayanan & Ankur Handa & Roberto Cipolla.
"SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling".
In: *arXiv preprint arXiv:1505.07293* (2015).

[32] Vijay Badrinarayanan & Alex Kendall & Roberto Cipolla.
"SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation".
In: *Computing Research Repository (CoRR)* abs/1511.00561 (2015).
URL: http://arxiv.org/abs/1511.00561.

[33] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre.
"Semantic Segmentation of Earth Observation Data Using Multimodal and Multi-scale Deep Networks".
In: *Computing Research Repository (CoRR)* abs/1609.06846 (2016).
URL: http://arxiv.org/abs/1609.06846.

[34] Alex Kendall. *SegNet*. 2015. URL:
http://mi.eng.cam.ac.uk/projects/segnet/images/segnet.png
(visited on 04/10/2017).

[35] Yuri Boykov & Olga Veksler & Ramin Zabih.
"Fast Approximate Energy Minimization via Graph Cuts".
In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.11 (Nov. 2001), pp. 1222–1239. ISSN: 0162-8828.
DOI: 10.1109/34.969114.
URL: http://dx.doi.org/10.1109/34.969114.

[36] Vladimir Kolmogorov & Ramin Zabin.
"What energy functions can be minimized via graph cuts?"
In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2 (Feb. 2004), pp. 147–159. ISSN: 0162-8828.
DOI: 10.1109/TPAMI.2004.1262177.

[37] Yuri Boykov & Vladimir Kolmogorov. “An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9 (Sept. 2004), pp. 1124–1137. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.60.

[38] Yuri Boykov & Olga Veksler & Ramin Zabih. “Efficient Approximate Energy Minimization via Graph Cuts.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Nov. 2001).

Appendix A

The Setup

Here follows the directions to setup and run scripts and programs, which were used throughout the thesis. The setup is divided into three parts: the modelling, synthesizing and classification part. The modelling and synthesis are in Matlab directory, whereas the classification part is under Python directory.

A.1 Modelling

The configuration parameters can be found in a Config file, where we setup all the needed parameters to begin the modelling of clouds or synthesizing from the modelled clouds. The modelling script is named ModellSetup. In the script, we specify the ground truth cloudless image, and the reference image, from which clouds will be modelled. The option to influence modelling are in Config file. The procedure outputs: metaClouds structure, evaluation criterion for each step, approximated image and a residual image after the approximation is done. It is necessary to visit ApproxClouds in order to change the approximation technique. The optimization procedure is composed from MinimizeMetaball, MinimizeMetaball2, OptimizeMetaball.

A.2 Synthesize

The synthesizing procedure uses the same configuration file as the modelling part. The image generating files are CreateTestImages and CreateTestCloudsImages. In those files we need to specify ground truth cloudless image again, due to the possibility of differentiation between modelling and synthesizing.

A.3 Classification

There is a dependency to build a Caffe library, due to the development of our classifier in the Caffe library. We have used a fork of Caffe library, that includes Alex Kendall's unpooling layer, this fork can be downloaded by cloning a repository with a branch *upsample* from

$$https://github.com/nshaud/caffe/tree/upsample$$

.

There is a need to compile the Caffe library with *make* and $Makefile.config$, however Caffe depends on other libraries: CUDA is required for GPU mode. BLAS via ATLAS, or MKL, or OpenBLAS, Boost library with version greater than 1.55, protobuf, glog, gflags and hdf5 libraries. Our code is scripted in python, thus python library of at least 2.7 is needed.

The weights of the network, which we used as an initialization are released under Creative-Commons BY-NC-SA. Caffe is released under the BSD 2-Clause license.

The training can be executed with a command python training.py –niter number of iterations –snapshot path to save snapshot –init path to initialization weights, or instead of initialization we can restore the weights from solverstate to resume learning, where we need to change –init to –restore. The training.py file is located inside Python folder with a Config file, which enables the changes of dataset. The evaluation of dataset is processed in a file Evaluation.py, where we need to set up required variables. The inference on image is produced in Tests.py. The dataset is created inside Utilities.py. All the python files are configurable with appropriate Config.py file.