Classification of satellite images using Convolutional Neural Networks

Sanne Ingvarsson, Susanna Larsson, Anna Birgersson, Jakob Grönlund, Linus Eriksson
CDIO-project in TSBB11
Linköping University
Wednesday 19th December, 2018

Abstract

This technical documentation describes the work and results of the CDIO-project "Classification of satellite images using Convolutional Neural Networks". The aim of the project was to detect and label airports in satellite images using machine learning. Two different neural networks were used and evaluated in order to achieve this. The group have found that the networks perform somewhat differently, where DeepLabV3+ reached the best performance with an F1-score of 98.3% for non airport pixels and 75.3% for airport pixels. The project group conclude that airports are a challenging target to classify by using neural networks. This may be a result of the varied appearances of airports around the world, as well as the look of airports tends to have a similar appearance as urban surroundings.

1. Introduction

Accurate geodata is required when developing tomorrows telecom network, where an increased demand are seen in several fields such as video, 5G and IoT solutions. By using geodata, it is possible to plan and design telecom networks with optimal performance. Telecom networks are often overloaded in crowded areas, and by localizing social structures that contains a lot of people, it is possible to distribute resources more efficiently.

This paper describes how deep convolutional networks can be used for classification of social structures in satellite images. The project aims to classify airports in satellite images with two different neural networks and to further evaluate the both of them. The project originates from Carl Sundelius master thesis project at the company Vricon, and it aims to develop on his thoughts and findings [1].

1.1. Parties

The group consists of students from the course TSBB11. The supervisor from LiU is Gustav Häger and the data and assignment are given from Vricon. Vricon also provided guidance via Carl Sundelius.

	Description
1	The product should take inputs from five channels (R, G, B, NDVI, DSM).
2	The system should be able to handle the image format ".tif".
3	The system should be able to classify at least one new class, preferable airports, from satellite images. The F1-score should be at least 0.80.
4	The results should be evaluated quantitatively using F1-score and qualitatively by using visual evaluation.
5	At least two networks with different architectures should be compared. If there is time further architectures should be implemented and compared.

Table 1. Requirements.

1.2. Aims

The aim of this project is to train and evaluate two different neural networks with the purpose to classify airports from satellite images. The goal is to efficiently process satellite images and for all pixels determine if they correspond to airports or non-airports. The formal requirements are listed in table 1.

1.3. Definitions

- CNN: Convolutional neural network
- NDVI: Normalized difference vegetation index.
- DSM: Digital surface model
- RGB: Red, Green, Blue images

1.4. Restrictions

Throughout the project, data provided by Vricon will be used for training, test and validation. The project will be examining and using the networks ResNet50 and DeepLabV3+. Also a simple test network will be implemented and used as a reference of the training result. Two classes will be used, airports respectively non-airports. The

implementation will be done using the Keras api for Tensorflow, coded in Python. The end product will be command-line based and thus have no Graphical User Interface.

2. Theoretical background

This section aims to give the reader an overview of image semantic segmentation and convolutional neural networks. In 2.5 the two implemented networks are described in detail.

2.1. Convolutional Neural Network - CNN

The fundamental concept of a neural network is to mimic the neurons in a biological brain. Thereby, a neural network consists of neurons that are fed an input feature, which is represented by a number. The feature is then multiplied with the weight of the neuron, and the result is sent to the final output stage of the neural network. There, the results from all neurons are summed, and if a certain threshold value is reached, the activation function will spark. This is the basic concept of a binary classifier, a perceptron [2]. The network can then be trained to perform different tasks dependent on the training data. This data is pre-labeled, where each label corresponds to a previously known class. The weights the neurons are assigned in the net may then be randomized, and the network fed the training data. An error is formed, based on the difference in given classes from the network, and true classes from the training data. The error in classifying may then be used in a gradient descent algorithm, solving for the weights to minimize the error. In that way, the neural network can be trained to perform certain tasks, such as classifying if an image portraits a cat or not [2].

The neural network can from here be expanded to solve further tasks by letting the output of one layer of neurons connect as the input to another layer [3]. This creates a multi-layer perceptron, capable of performing tasks the original perceptron design is incapable of [4]. Letting the output of every neuron in a layer connect as input to every neuron in the next creates a fully-connected network, which can be efficiently trained to perform many tasks [3, 5].

In image classification and semantic segmentation the standard methods for extracting features has for a long time been hand-crafted algorithms, such as binary operations or edge detection. These features are then fed to neural networks trained to use them as input to perform its task [6, 7]. To avoid the limitations of these hand-crafted algorithms, a proposed alternative is to instead train the network to perform the feature extractions [7, 8]. This has shown to outperform traditional, fully-connected networks [9, 10]. The features are extracted by using layers that are locally connected, meaning that instead of letting the output from every neuron in one layer be fed to the input of every neuron on the next, only ”spatially close” connections are formed. This ensures that the feature of one corner of an image is not affected by another corner. The resulting weights are thus more similar to a 2D convolution kernel. This further explains the feature-extraction, since a 2D convolution utilizes spatial information in an image, which is considered important in image classification and semantic segmentation [6].

2.2. Semantic segmentation

Semantic Segmentation describes the art of image classification on a pixel level where one aims to assign each single pixel with a certain class label [11]. The input is a natural image and the output a similar picture but with a desired number of classes labeled, see figure 1. Common approaches to solve this task was for a long time to use for example Random Decision Forests [12] or Support Vector Machines [13], although the progress of deep Convolutional Networks for the last few years have heavily outperformed these previous state-of-the-art solutions. By using the architecture of a deep CNN but replace the top layer with fully convolutional layers it is possible to predict classes in an image from an image of arbitrary size [11].

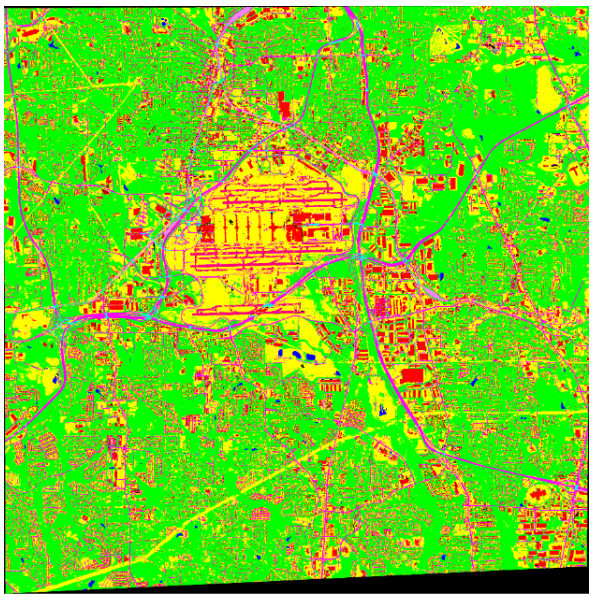


Figure 1. Previously classified image over Atlanta Airport where different terrain are given different color codes

2.3. Residual net

An intuitive solution when tasked with difficult machine-learning problems is to simply add more layers to the network, since a deeper neural network should in theory be better. However, it is shown that an increased depth of the network may result in a decline of accuracy in the performance [14, 15]. This behaviour has no correlation to

over-fitting neither over-training of the network, but rather appears to be an effect of the structure of the network itself. This seems to happen because of the network incapability to create so called *identity layers*, simply layers with the weight 1 and therefor with an input identical to its output [14].

Assuming that $y = F(x)$ is the correct mapping of input x to output y for a layer, let $\hat{F}(x) = F(x) + x$. Then it is suggested to use $F(x) = \hat{F} - x$ as the mapping for the layer [14]. This rewritten mapping is believed to be easier to optimize than the original one.

Thus, to achieve superior accuracy, residual nets employ "skip" steps, where the input from a layer number n is sometimes forwarded to the output from layer number $n+1$, effectively letting layer $n+1$ have the output $F(x)+x$. This behaviour can be seen in figure 2

2.4. F1-score

The F1-score is an evaluation method that considers both the precision and recall of the model and returns a weighted average of these, according to equation 1 [16].

$$\mathrm{F1-score} = 2 * \frac{precision * recall}{precision + recall} \tag{1}$$

A F1-score is obtained for each class and the precision and recall contribute equally to the score.

The precision is calculated as the number of true positives divided by number of total positives (both true and false). This tells us how many of the predicted positives that are actually correct and gives a measurement of the networks ability to not label a sample as positive when it is in fact negative[16].

The recall is calculated as the number of true positives divides by the number of total actual positives (both true positives and false negatives). This is a measurement of the networks ability to find all positive samples [16].

2.5. Network descriptions

This section aims to give the reader a more thorough understanding of where the used network originates from and describe its functionality.

2.5.1 ResNet50

Resnet50 is a residual network that in 2015 won the ILSVRC challenge for Large Scale Visual Recognition. It is a development of the previous deep CNN AlexNet that won the same challenge in 2012 [17] and VGGNet that won the challenge in 2014. During the challenge, participants main focus was to come up with new CNN architectures that could outperform the existing ones. VGGNet was at its time both deeper and bigger than its ancestor with 16 respectively 19 layers, which at the time was considered to be very deep [18]. ResNet, with an architecture of up to 152 layers, was presented the following year as a response to the proven theory that deeper network performed better. The number of parameters could be reduced thanks to the use of global average pooling rather than fully connected layers. Apart from the more accurate prediction, a problem with such deep network aligned with overfitting. He et al, the creators of ResNet50, introduced the residual blocks as skip connections between layers as illustrated in figure 2 to get around this problem [14].

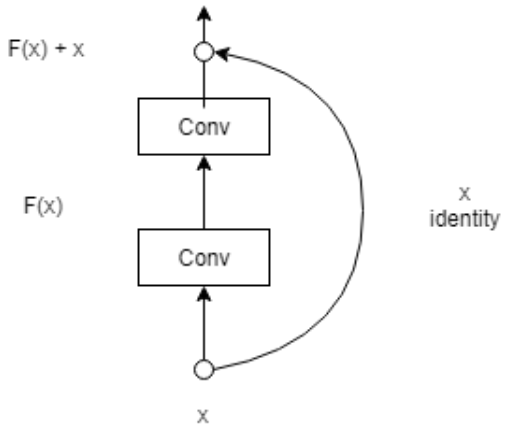


Figure 2. Residual skipping, as described in 2.3

ResNet allows an input of arbitrary size and downsamples the output from each layer until it in the final layer equals the number of classes to predict which in this case solely equals a binary classification [14].

2.5.2 DeepLab

Deeplabv3+ is based on the previous DeepLabv3 but with a simple decoder module to improve the segmentation along object boundaries [20]. DeepLabv3 applies Atrous Spatial Pyramid Pooling (ASPP), which is several parallel convolutions with different pooling rates, see figure 3 [20, 19]. This allows for capturing contextual information at multiple scales, which give much information of the semantics in the last feature map. The result does however lack detailed information of object boundaries due to the down sampling of the image along the neural network [20].

To improve segmentation at object boundaries DeepLabv3+ employes ASPP with a encoder-decoder structure. Figure 4 below shows the ASPP module, Encoder-Decoder structure

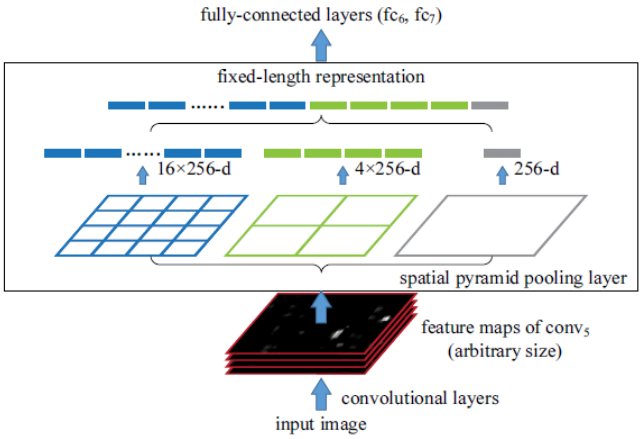


Figure 3. Spatial pyramid pooling [19]

and the Encoder-Decoder together with ASPP. Encoder-decoder networks does typically contain one encoder module that reduces the feature map with convolution layers and pooling in order to extract semantic information. To receive a result with the same resolution as the input the decoder gradually recover the spatial information to the encoded image [20].

The segmentation is performed using an adapted Xception model where depthwise separable convolution is applied on both ASPP module and decoder module. Deeplabv3+ is also using pre-trained weights from the PASCAL VOC 2012 dataset to boost the perfromance of the model [20].

3. Method

This section aims to give an overview of the method used in the project. The group was given a framework from Vricon with useful functions. Provided functions were for example creating and saving models, training locally and on Google Cloud. Further the group had to implement networks that was suitable for this frame as well as generating functions for prediction and evaluation.

3.1. Preprocessing of data

The satellite images the group received from Vricon were in the format of tif-images (Tagged Image File) and were of three different types. One image contained the RGB channels, one contained the NDVI and one contained the DSM. The images had various sizes but all of them were quite large (for example 7054x7814 pixels) and were thus too large to be used directly as input to the networks. It was also preferable to arrange the data into numpy arrays containing certain layers so that it would work properly with the training code that the group was given. Because of this, preprocessing of the images were necessary. The input images were chosen to contain seven different layers; R, G, B, NVDI, normlized DSM, labels and a weight mask. Below the different preprocessing parts are described briefly.

3.1.1 Normalize DSM

The given DSM tif-images were not normalized and also had values of approximately -32800 in pixels were data was missing. To not let this interfere and mess up the training, zeros were placed in these pixels. Then lowest point in the tif-image was set to zero and the other pixels obtained values relative to this lowest point. In this way the image height-values were normalized between 0 and 1.

3.1.2 Label airports

Since there was not any images in which airports had been labeled the group had to label the airports in the tif-images manually. This was done by using the image editor program GIMP2. The airports were marked in a red color and everything that was not an airport in black. Then the images were modified so that airport pixels obtained the value 1 and non-airport pixels the value 0. These images were saved as tif-images.

3.1.3 Creating weight mask

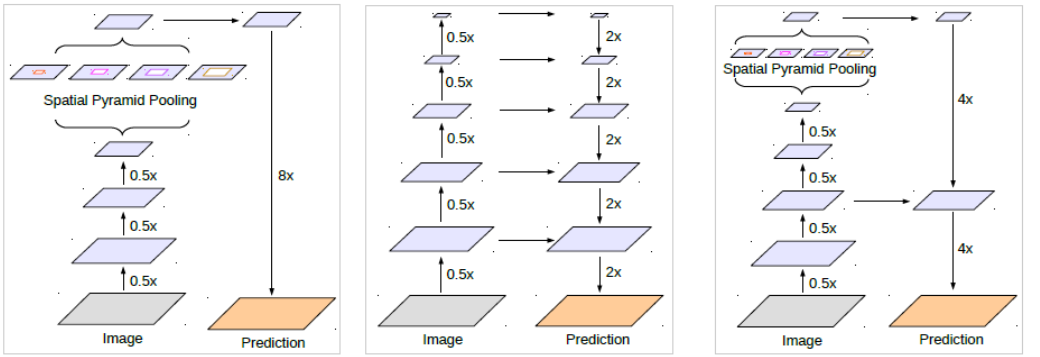
The ratio between the two classes (airport and non-airport) vary widely between the images. If the test set is very unbalanced (i.e a ratio of 10:100000) the network could get a good accuracy by only predicting non-airport for all pixels, although this is not desirable since the interest lies in classifying the airports. To overcome the problem (or at least make the problem less significant) a weight mask that weights the importance of every pixel was created to be used during training. The pixels containing airports got the value 1 and the value of the non-airport pixels was set to a value according to the following:

$$W_{non-airport} = \frac{\text{number of airport pixels}}{\text{number of non} - \text{airport pixels}} \quad (2)$$

Pixels that were not supposed to be used in the training were set to zero. It is the ratio between the pixels in the weight mask that are important to make the training more efficient.

3.1.4 Splitting into smaller patches

As mentioned earlier the tif-images were too large to be used directly as input data. After obtaining the normalized DSM, labeled tif-image and weight mask the images were converted into numpy arrays and split into 512x512 patches. Then the seven different layers were added together and only the patches containing airport or parts of an airport was saved. Each of these patches then got a size of 512x512x7 and were ready to be used as input to the



(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv.

Figure 4. Encoder-Decoder and ASPP [20]

networks.

The training data were then added to the storage on Google cloud were 2/3 was used for the training and 1/3 for the validation. Some of the images was saved to be used as test to evaluate the result.

3.1.5 Augmentation of data

A large amount of training data is usually required in order for the network to learn efficiently [21]. In this case the group had a limited amount of labeled training data so data augmentation became relevant. Each image was randomly rotated 90, 180 or 270° right before they were chosen for training or validation. The network was then feed the several different rotated versions of the same image. The rotational difference was perceived by the network as new data, and thus more training data was ”created”.

3.2. Implementation of Network

This section describes the implementation of the networks and how they have been tweaked to fit the set requirements. Two different networks were supposed to be tested where the system should be able to use .tif images as input and output. Each network should be able to take a patch of 512x512 pixels with five channels as input.

3.2.1 ResNet50

The ResNet50 model used was taken from Keras Applications, which is a library with multiple deep learning models available together with pre-trained weights [22]. Parts of the model was pre-trained on weights from ImageNet, in order to achieve better initialization for the weights. The ImageNet weights available were ment to be used with a network taking 224x224 RGB images as input and clasify the images. Therefore, some modification to the network was neccesary. This was done to achieve an architecture which supported the desired input and output (that is, 512x512x5 and 512x512x2). To achieve this, two ResNet50 models were created. The first of them contained pre-trained weights from ImageNet on all layers. The second had a removed input-layer, which allowed for an arbitrary input size, and randomized weights. The weights from the first model was then copied onto the second model, with the exception for the non-existent input-layer. This effectively creates a ResNet50 model with pretrained weights for the RGB-channels. The output from the final convolutional layer from this model was then taken and connceted to a convolutional layer, outputting a 512x512x2 tensor. This was finaly fed to a softmax-layer, performing the final pixel-wise classification.

3.2.2 DeepLabV3+

The DeepLabV3+ model was found as Keras compatible open source and was retrieved from GitHub [23]. The model was created with pre-trained weights from the PASCAL VOC dataset. The pre-trained weights were trained on 3 channel images (R, G, B) and therefore the model needed to be modified to fit the images with 5 channels.

To handle the problem with channels, the first layer in the model was modified by changing the kernel size and name of the layer. By loading the PASCAL VOC weights by name they were loaded into all layers except the first one. The weights of the first layer was instead initiated with random weights.

Since a cross entropy loss function was used the output of the model needed to be scaled between $[0, 1]$. To solve this a final softmax activation function was added. Finally the output was reshaped to fit the weight mask described above.

3.2.3 Simple Net

A small randomized Neural Network was created as a reference net and was used to see whether DeepLabV3+ and ResNet50 performed better than solely randomized convolutions layers. The simple network contained only 10 convolutions layers.

3.3. Training

The training was performed using a Google cloud compute engine with GPU support.

As the learning rate of a network can greatly affect its performance, finding the optimal learning rate for each network was deemed crucial. This was done by letting the network train 10 epochs with different learning rates, ranging from 10^{-3} to 10^{-9}. The validation loss from each training run was then noted and the learning rate that generated the lowest validiation loss was selected to run the final training with 400 epochs or until the model could not improve any more. By using early stopping the training was stopped when the validation loss was not marginally improved between epochs and the model was as good as possible. The used batch size was 5 which was the highest possible due to a limited amount of VRAM.

Two training sets were created, one with and one without the weight mask. Both models were trained with both sets to evaluate how the weight mask affected the final results.

3.4. Prediction

The trained model was used to perform classification on new images that had not been used in training. The used implementation for this was a $predict$ function provided by Keras [24]. $predict$ returns an 512x512 array with same depth as the number of classes, i.e. 2 (one for airport and one for not airport). The sum of the values in both layers are always 1 in each pixel. By selecting the highest value of the two layers each pixel gets their classification.

To be able to use the Keras implementation the first step was to transform the .tif images into a 5-layers array as in the preprocessing step. The array was then divided into patches with size 512x512 pixels and each of the patches were predicted one by one. The predicted result of each patch was then added to an array with same shape as the .tif images but with one layer.

3.5. Evaluation

The evaluation method used to evaluate the networks is the F1-score. This method was chosen since it takes both the precision and recall into account, which was considered important to get a fair and sensible evaluation.

Since two F1-scores was obtained, one for airports and one for non-airports, an average F1-score was calculated as the average of the two scores. This was done to get a measurement of the overall performance of the networks.

4. Results

This section presents the result of the project.

4.1. Learning rates

The results from the learning rate test are displayed i the tables below. Tabel 2 shows the results from the DeeplabV3+ implementation, tabel 3 shows the results for the ResNet50 implementation and the results for the simple test network are shown in tabel 4.

Learning Rate	Validation Loss
10^{-3}	0.67421
10^{-4}	0.31869
10^{-5}	0.32248
10^{-6}	0.55617
10^{-7}	0.56366
10^{-8}	0.65102
10^{-9}	0.6498

Table 2. Validation loss with varying learning rate for DeepLab

Learning Rate	Validation Loss
10^{-3}	0.34905
10^{-4}	0.2595
10^{-5}	0.49204
10^{-6}	0.4125
10^{-7}	0.611
10^{-8}	0.64025
10^{-9}	0.6493

Table 3. Validation loss with varying learning rate for ResNet50

Learning Rate	Validation Loss
10^{-3}	4.345
10^{-4}	4.345
10^{-5}	4.339
10^{-6}	3.522
10^{-7}	4.325
10^{-8}	6.056
10^{-9}	6.264

Table 4. Validation loss with varying learning rate for the simple test network

As can be noted in table 2 and table 3, the lowest validation loss for both implementations was obtained when the learning rate was set to 10^{-4}. This learning rate was thus deemed optimal and used in further training. The test also

showed that the simple test neural network performed way worse than the designed ones.

4.2. Prediction and evaluation of DeepLabV3+

The results of the predictions with the DeepLabV3+ model of three different images are shown in the figures below. The pink color corresponds to the predicted airport pixels. Random rotation for data augmentation was used.

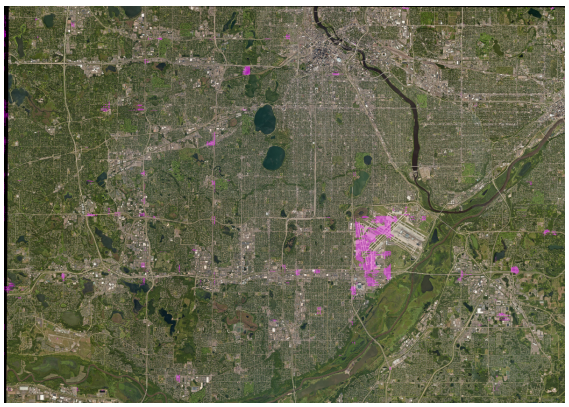


Figure 5. Minneapolis predicted with DeepLabV3+, no weight mask used.

	No airport	Airport
No airport	11826059	304909
Airport	90452	616468

Table 5. Confusion matrix from prediction of the image in figure 5

The F1-score for the prediction in figure 5 is 99.0 % for non airport pixels and 31.9 % for airport pixels, with an average F1-score of 65.5 %.

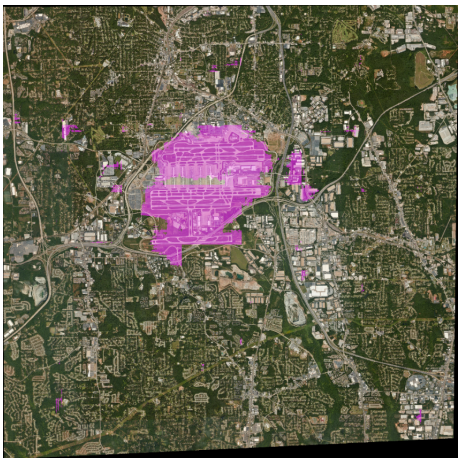


Figure 6. Atlanta airport predicted with DeepLabV3+, no weight mask used.

	No airport	Airport
No airport	11825736	312400
Airport	90775	616145

Table 6. Confusion matrix from prediction of the image in figure 6

The F1-score for the prediction in figure 6 is 98.3 % for non airport pixels and 75.3 % for airport pixels, with an average F1-score of 86.8 %.

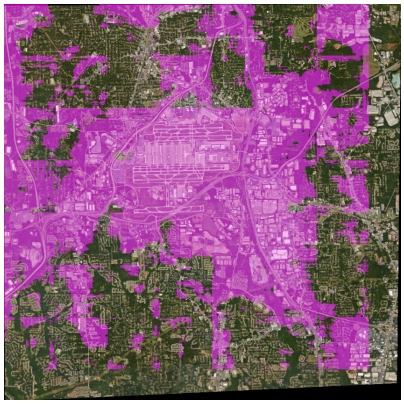


Figure 7. Atlanta airport predicted with DeepLabV3+, weight mask used.

	No airport	Airport
No airport	5443907	6472604
Airport	28388	900157

Table 7. Confusion matrix from prediction of the image in figure 7

The F1-score for the prediction in figure 7 is 62.6% for non airport pixels and 21.7% for airport pixels, with an average F1-score of 42.1 %.

Figure 8. Singapore airports predicted with DeepLabV3+, no weight mask used.

	No airport	Airport
No airport	52310497	2127110
Airport	29493	58852

Table 8. Confusion matrix from prediction of the image in figure 8

The F1-score for the prediction in figure 8 is 97.9 % for non airport pixels and 5.2 % for airport pixels, with an average F1-score of 51.6 %.

4.3. Prediction and evaluation of ResNet50

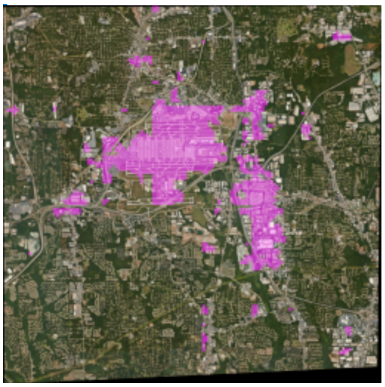


Figure 9. Atlanta airport predicted with ResNet50, learningrate 0.001 no weight mask used.

	No airport	Airport
No airport	11205873	710638
Airport	245679	675698

Table 9. Confusion matrix from prediction of the image in figure 9

The F1-score for the prediction in figure 9 is 95.9 % for non airport pixels and 58.6 % for airport pixels, with an average F1-score of 77.3 %.

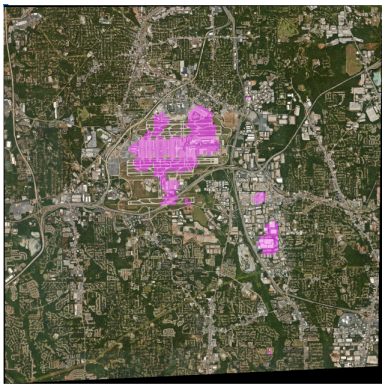


Figure 10. Atlanta airport predicted with ResNet50, learningrate 0.001 weight mask used.

	No airport	Airport
No airport	11848815	512913
Airport	67696	415632

Table 10. Confusion matrix from prediction of the image in figure 10

The F1-score for the prediction in figure 10 is 97.6 % for non airport pixels and 58.9 % for airport pixels, with an average F1-score of 78.3 %.

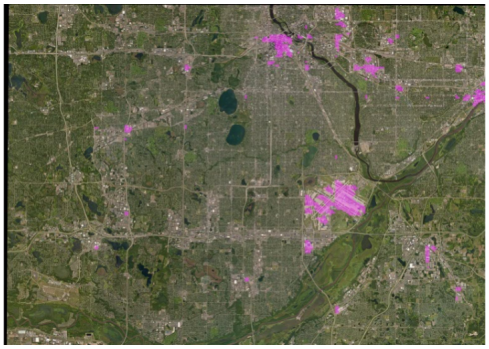


Figure 11. Minneapolis airport predicted with ResNet50, learningrate 0.001 no weight mask used.

	No airport	Airport
No airport	44827415	492760
Airport	527593	289576

Table 11. Confusion matrix from prediction of the image in figure 11

The F1-score for the prediction in figure 11 is 98.9 % for non airport pixels and 36.2 % for airport pixels, with an average F1-score of 67.6 %.

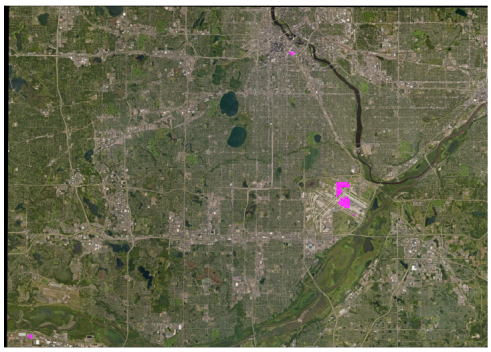


Figure 12. Minneapolis airport predicted with ResNet50, learningrate 0.001 weight mask used.

	No airport	Airport
No airport	45311092	1766714
Airport	461131	356038

Table 12. Confusion matrix from prediction of the image in figure 12

The F1-score for the prediction in figure 12 is 99.2 % for non airport pixels and 13.7 % for airport pixels, with an average F1-score of 56.5%.

Figure 13. Singapore airport predicted with ResNet50, learningrate 0.01 weight mask not used.

	No airport	Airport
No airport	52271382	68608
Airport	2120426	65536

Table 13. Confusion matrix from prediction of the image in figure 13

The F1-score for the prediction in figure 13 is 98.0 % for non airport pixels and 5.6 % for airport pixels, with an average F1-score of 51.8 %.

Figure 14. Singapore airport predicted with ResNet50, learningrate 0.001 weight mask used.

	No airport	Airport
No airport	51877166	462824
Airport	2105042	80920

Table 14. Confusion matrix from prediction of the image in figure 14

The F1-score for the prediction in figure 14 is 97.6 % for non airport pixels and 6.0 % for airport pixels, with an average F1-score of 51.8 %.

5. Discussion

Below follows a discussion regarding the performance of the two different networks and possible reasons why the results look like they do. Also future possible improvements of the system and networks are discussed in this section.

5.1. Labeling of data

The amount of labeled data is probably the main problem when it comes to classification using a neural network. The labeled data consist of totally 60 satellite images with airports where all airports have different shapes and sizes - some airports are bigger than others, some have more grass around/between the runways, they might have multiple runways or a single runway, they might be surrounded by many or almost no buildings at all, etc.

The group has labeled the data in GIMP2 and had sometimes problem to localize the right areas according to their appearances and had to use Google maps to find their locations. It was also hard to decide what should be included in the airport labeling and what should not. Since all group members took part in the labeling process it is also possible that what was labeled as airports varied depending on which group member performed the labeling. Further on, when labeling, the group found that it was often hard to see the difference between runway and big roads with houses surrounding it. It is possible that the models experienced the same difficulties.

Since both networks has several million parameters it would take a lot of time to train them from scratch and because of this pre-trained weights was needed. This means that the network has already learned some universal features and this will then speed up the training process. Even though the pre-trained weights are used it still seemed like the amount of training data was to small to obtain the desired results. One possible bad impact on the result cloud be that the first layer did have random start weights, which may have been badly initialized.

5.2. Discussion of result

The results of the predicted images shown in the result section are here discussed and analyzed, both for DeepLabV3+ and ResNet50.

5.2.1 DeeplabV3+

The network trained with the DeepLabV3+ model produced a good result on the Atlanta airport shown in figure 6. The good result of that prediction might be because the airport had a similar appearance as many of the other airports that was used in the training set. Another reason could be that the airport was large in size.

A bad prediction with the network is seen in figure 8 that illustrate Singapore airport. The reason for this was probably that the airport does not look similar to most other airports the network used for training. The surroundings of the runways was in this case brown and not green. To learn the network to find airports with different appearance like this, more training data with a larger variety of the appearances of airports are needed.

From the result of the predictions for both ResNet50 and DeepLabV3+ it is clear that the prediction of Atlanta airport got a higher average F1-score compared to the prediction of Minneapolis. A reason for this could be that the size of the airport matters and that larger airports are easier to find. By looking at the Minneapolis prediction that actually contains two airports, one larger that are easily seen and one smaller in the lower left corner, only the larger airport are predicted as airport.

5.2.2 ResNet50

As can be seen in figure 9 and figure 11, the ResNet50 implementation appears to perform acceptably on the Atlanta airport and the Minneapolis airport. This is attributed to the clear contrast between the airports and their surroundings, as well as their ”typical” appearance. This indicates that the ResNet50 implementation can perform well, but that it requires sufficient training data and is restricted to identifying airports with a certain ”look”. This is further illustrated in figure 13 and figure 14, where the airport in Singapore is clearly not detected, and that several other places instead are wrongly marked. This may indicate that more training data is needed to generalize the ResNet50 implementation to include airports such as the one in Singapore. A possible explanation of the currently poor performance of this airport is the lack of similar airports in the training data, that is, coastal airports with similarly colored surroundings.

5.3. Weight mask

A notable difference was found in performance between networks when weight masks was used and when it was not used. As seen in figure 4, it is clear that when DeepLabV3+ is trained with the weigh mask, the networks labels pixels as airports more aggressively than without the weight mask, which gives a worse result. This is in accordance with the theory presented in section 3.1.3. Since airport pixels have higher weights than non-airport pixels, the network predicts more of them. The weight mask would probably be more useful when predicting more classes than two. Worth noticing is that the learning rate seems to play an important role when one decides whether a weight mask should be used or not. A clear example of this is to be seen in figure 10, which even performed somewhat better than without mask. If the learning rate is to high it is common that the model diverges and predicts the highest weighted class, which in our case is the airport. This behaviour was also seen for the ResNet50 model, where the showcased results with weight mask solely could be reached with this certain learning rate.

The results from the ResNet comparison with and without weight mask is somewhat contradictory to what is mentioned in 3.1.3. This is since the weights are supposed to make sure that airport pixels are correctly labeled, caring less about the non-airport pixels. Based on the theory presented in section 3.1.3, the networks without masks should have created more false-negatives and less false-positives, since an easy way to achieve high accuracy is to think of everything as non-airports. This is, as mentioned, observed in the DeepLabV3+ networks. Since the observed result for ResNet50 points to the contrary, it is believed that more training data is needed to let the weights have effect, and that further research in general is needed.

5.4. Comparison between the networks

As seen in table 4 the simple test network performed way worse than the designed ones and the result can be seen as a passed sanity test for the project. The test result assured that the complex structure of ResNet50 and DeeplabV3+ actually added something to the segmentation process.

The results shows that the best performance from the DeepLabV3+ implementation is an average F1-score of 86.8 %. This is achieved on the Atlanta airport with learning rate 10^{-2} and no weight mask. This can be compared to the best performance of the ResNet50 implementation, with an average F1-score of 78.3 %. This is achieved on the Atlanta airport using a learning rate of 10^{-3} and a weight mask.

Based on these results, it can be argued that the

DeepLabV3+ implementation performs better than the ResNet50 implementation, since the achieved F1-scores are higher for both airports and non-airports. This is attributed to the fact that DeepLabV3+ is developed to perform semantic segmentation, while ResNet50 is developed to perform image classification. This is believed to cause DeepLabV3+ to perform better than ResNet50 given the same training data. This fact is believed to be a result of the groups tampering with ResNet50, and the made modifications to allow it to perform semantic segmentation. The modifications made by the group did not result in a network design of the same quality as DeepLabV3, a result of the groups inexperience and time-limit.

The fact that both implementations have a very low accuracy when applied to the Singapore airport is explained with a lack of training data. It is argued that more images, with similar background and environments as the Singapore one, would allow both networks to perform better.

As discussed in section 5.3, a difference between the networks is how they perform with weight masks and without. DeepLabV3+ clearly supports the theory presented in section 3.1.3, labeling more pixels as airports with the weight mask than without. The use of weight mask on DeepLabV3+ lowered the networks total accuracy. ResNet50, on the other hand, labeled less pixels as airports with the weight mask, and achieved a higher total accuracy with the weight mask. This inconsistency is deemed interesting, and worth further investigation, based on the fact that the networks behaved similary otherwise.

5.5. Possible improvements

One possible improvement could be to postprocess the predicted images. As seen in the results the predicted images contains many small areas that are marked as airports but that are not in fact airports. Since airports usually have a large size it would be possible to remove predicted airports that are only a few pixels or that not are large enough to airports. This would probably give an overall better F1-score and accuracy, but could also risk to accidentally remove very small airports or runways from the predicted images

Future work could also be to not calculate the amount of detected airport pixel, but rather number of airports and an estimation of where they are. Airports would then be found where the model has detected a cluster of airport pixels. That would require post-processing of the result similar as described above, where clusters that are to small or to big to be an airport are removed.

One other improvement could be to use more rotation angels and use different scaling of the images when doing the data augmentation. To solve the main problem more labeled airports would be the best solution though.

6. Conclusion

The conclusion that can be drawn from this project is that airports are a challenging type to classify, possibly because of their variety in appearance as well as similarity to other social structures. The experiment does however show that it is possible to train these models to find larger airports, but with an insufficient reliability. The best average F1-score for DeepLabV3+ achieved a result higher than the required of 0.80, while the ResNet50 implementation did not. To get a more accurate classification it is obvious that a much large set of training data is required, not at least when training a neural network that has millions of parameters.

References

[1] Carl Sundelius. Deep fusion of imaging modalities for semantic segmentation of satellite imagery, 2018.

[2] Frank Rosenblatt. The perceptron a percieving and regocnizing automatton. *Cornell Aeronautical Laboratory INC*, 1957.

[3] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015.

[4] Seymour A. Papert Marvin Minsky. *Perceptrons: An Intrduction to Computational Geometry*. The MIT Press, 1969.

[5] Paul John Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974.

[6] Yoshua Bengio Patrick Haffner Yann LeCun, Léon Bottou. Gradient-based learning applied to document recognition. *IEE*, 1998.

[7] Yoshua Bengio Yann LeCun. Convolutional networks for images, speech and time-series. *The handbook of brain theory and neural networks*, pages 255–258, 1998.

[8] J. S. Denker D. Henderson R. E. Howard W. Hubard L. D. Jackel Y. KeCun, B. Boser. Backpropogation applied to handwriten zip code recognition. *Neural Computation*, 1989.

[9] H.P Graf J. S. Denker Y. LeCun D. Henderson O. Matan R. E. Howard L. D. Jackel, B. Boser. Vilsi implementations of electronic neural networks: An example in character recognition. *IEE*, 1990.

[10] Yusuke Mitari Yuji Kaneda Masakazu Matsugu, Katsuhiko Mori. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 2003.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[12] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[13] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[14] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *CVPR*, 2015.

[15] Jian Sun Kaiming He. Convolutionan neural networks at constrained time cost. *CVPR*, 2015.

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

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European conference on computer vision*, pages 346–361. Springer, 2014.

[20] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[21] Luis Perez Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *ArXiv*, 2017.

[22] François Chollet et al. Keras. `https://keras.io`, 2015.

[23] bonlime. Keras implementation of deeplabv3+.

[24] Keras. The sequential model api, 2018 (accessed 27.11.2018). https://keras.io/models/sequential/.