

Unterschrift des Betreuers

DIPLOMARBEIT

Use of Pictures from Social Media to Assess the Local Attractivity as an Indicator for Real Estate Value Assessment

ausgeführt am

Department für Geodäsie und Geoinformation (E120)

FG Geoinformation

Der Technischen Universität Wien

Unter Anleitung von

Priv. Doz. Dipl. Ing. Dr. techn. Gerhard Navratil

durch

Christopher Kmen, Bakk. techn.

Matr.-Nr. 0826138

Hackenberggasse 29/10/1, 1190 Wien

Eidesstattliche Erklärung

Ich erkläre an Eides Statt, dass ich die vorliegende Master Thesis selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und alle Quellen wörtlich oder sinngemäß entnommene Stellen als solche kenntlich gemacht habe.

Wien, am 07.12.2017

Unterschrift

Abstract

In recent years there has been a massive increase in the production and collection of data [Goodchild 2007]. Especially in the field of social media an overflowing quantity of pictures is produced. Therefore, the question is raised, if spatial models could be derived from these images. Or, in other words, is it possible to use social media data for spatial and/or semantic purposes?

In recent studies by Hochmair [2009] and Alivand [2013] it was found that people tend to make more pictures in places which appear more attractive than in those which seem less appealing. Other Studies (Brunauer et al. [2013] and Helbich et al. [2013]) come to the conclusion that those areas that appear more appealing have higher real estate prices.

This study will link all these components together. Images are collected from social media and classified based in their focus - social interaction or documentation of the surrounding. Images in the later case will be used for further analysis. A neural network will be used for classification. As area for the study Vienna is chosen.

In the next step another big amount of social media images with geo location features is gathered and filtered with the newly trained neural network. Then the location information of the valid images is stored. Out of these data a heat map is created, with the density of the images taken as indicator.

For the validation of the created model the company DataScience Service GmbH compares the heat map with their real estate price model to see if there is a link between social media output and real estate prices.

Abstract

In den vergangenen Jahren wurde ein massiver Anstieg bei der Generierung und Speicherung von Daten beobachtet [Goodchild 2007]. Besonders im Social Media Bereich konnte ein signifikanter Anstieg bei der Erstellung von Bildern verzeichnet werden. Aus dieser präsenten Entwicklung heraus, stellt sich die Frage, ob diese Daten herangezogen werden können um raumbezogene Modelle daraus zu extrahieren und für semantische Fragestellungen heranzuziehen.

Studien von Hochmair [2009] und Alivand [2013] haben gezeigt, dass in Umgebungen welche als besonders schön empfunden werden mehr Bilder entstehen als in weniger attraktiven Gegenden. Außerdem wurde belegt, dass Immobilienpreise in Diesen deutlich höher sind (Brunauer et al. [2013] und Helbich et al. [2013]).

In der folgenden Arbeit wird eine Verbindung zwischen diesen erwähnten Studien hergestellt. Dafür werden zahlreiche Bilddateien akquiriert und unterschieden ob das Umfeld oder die soziale Interaktion, Grund für die Aufnahme ist. Zur Klassifizierung wird ein neurales Netz verwendet. Das Testgebiet erstreckt sich über Wien.

Des Weiteren werden Positionsinformationen der validen Bilder gespeichert, um anschließend daraus eine „Heat-Map“ zu erstellen. Als Indikator dient die Fotodichte der gemachten Bilder.

Um herauszufinden ob eine Verbindung zwischen existierenden Social Media Bildern und Immobilienpreisen besteht, wird das geschaffene Modell von der Firma DataSciences GmbH mit ihren Immobilienpreismodell verglichen.

Danksagung

Ich möchte mich bei meinem Diplombetreuer Dr. Gerhard Navratil für seine Hilfe und Unterstützung während der Entstehung meiner Abschlussarbeit bedanken. Auch für seine zahlreichen Tipps und dafür, dass er immer für mich Zeit gefunden hat.

Weiters danke ich meinem Freund Dino Valic für seine Hilfe bei der technischen Umsetzung und meinem Freund und Kollegen Sebastian Flöry.

Ein weiterer Dank gebührt der Firma DataScience GmbH für die Validierung meines Modells, wodurch ein Realitätsbezug hergestellt werden konnte.

Großen Dank möchte ich meinen Eltern und meinen Freunden Bruno, Fabian, Kevin und Marco aussprechen, die mich während meines gesamten Studiums unterstützt haben.

Zuletzt möchte ich mich bei meiner Freundin Vera bedanken, die mir immer zur Seite steht, mich während meiner Studienzeit unterstützt hat und immer verständnisvoll war. Ich möchte ihr und unserer gemeinsamen Tochter Valerie, diese Arbeit widmen.

1	INTRODUCTION	1
1.1	Motivation	1
1.2	Feasibility Analysis & Research Questions.....	1
1.3	Papers Related to this Work.....	2
1.4	Approach	2
2	BACKGROUND	4
2.1	Photo platforms	4
2.1.1	Instagram	4
2.1.2	Flickr.....	5
2.1.3	Location-Metadata.....	7
2.2	Deep Learning & Neural Networks.....	9
2.2.1	Convolutional Neural Network	10
2.2.2	Inception V3	11
2.2.3	TensorFlow.....	12
2.3	QGIS	14
2.3.1	Vector data.....	14
2.3.2	Raster data	15
2.3.3	Coordinate System	15
3	METHODOLOGY	16
3.1	Hardware and OS:	16
3.2	Valid images.....	16
3.2.1	Invalid classes.....	17
3.2.2	Valid class.....	23
3.3	GNSS VS Geotag	24
3.4	CNN & TensorFlow	26
3.5	Data output & GIS	26
4	WORKFLOW	28
4.1	Creating a training set	28
4.2	Training of the Neural Network.....	30
4.3	Testing the model	32
4.4	Data for the model.....	34

4.5	Classify images	38
4.6	Creating the Heat map	39
4.6.1	Load the data in QGis.....	39
4.6.2	Preparation of the data.....	42
4.6.3	Heat map tool	43
4.7	Challenges.....	44
4.7.1	Testing the model	44
4.7.2	Connectivity issues.....	45
5	RESULTS	46
5.1	Image Statistics	46
5.1.1	Downloaded data.....	46
5.1.2	Classified images.....	46
5.1.3	Point data for the model.....	46
5.2	QGIS.....	49
5.2.1	1 st district “InnereStadt”	51
5.2.2	3 rd district “Landstraße”	52
5.2.3	13 th district “Hietzing”	54
6	VALIDATION	56
6.1	Verification set up.....	56
6.2	Reality claim of the photo variable	58
7	SUMMARY AND FURTHER RESEARCH.....	60
8	REFERENCES	62
9	ILLUSTRATION DIRECTORY	64
10	ATTACHMENT	69
10.1	Image derivations of the CNNs	69
10.2	Scripts.....	71
10.2.1	Downloading script	71
10.2.2	Test script.....	76
10.2.3	Classification script.....	78

1 Introduction

1.1 Motivation

In recent years Web 2.0 has gained more and more influence, which is directly linked to an increase in Volunteer Geographic Information (VGI) [Goodchild 2007]. Especially in terms of social media the amount of produced and collected VGI data significantly increased in the last decade. It might appear that these data cannot be used for spatial computations, but since the rise of Global Navigation Satellite Systems (GNSS) featured devices most of the data produced have excellent geo references and therefore could be employed for various computations.

In recent studies [Hochmair 2010, Alivand & Hochmair 2013] it was found that the amount of pictures taken in a certain area could be linked to the beauty of this area. This local attractiveness then could be used as an indicator for real estate prices. The assumption was created that by extracting images from social media platforms with a link to a certain area it would be possible to create a model. The density model build from social media images would show image hotspots and therefore areas which are perceived as beautiful. This model then could be used as an indicator for real estate prices. Helbich et al. [2013] and Brunauer et al. [2013] showed that the beauty of an area is an indicator for the real estate prices achieved in this place. To go one step further the proposition is claimed that these areas which appear nicer are more expensive in terms of real estate prices.

1.2 Feasibility Analysis & Research Questions

The aim of this study is to examine if social media VGI could be used to extract an attractiveness indicator for real estate prices. Furthermore, the study should develop techniques for automatic classification of photo VGI data and extracting their position information on bigger scales.

The research topics of this study are:

- How valid images could be found and extracted
- How to automatize the classification process
- How to display the classified data
- Is there a link between social media output and real estate prices

The area of interest in this study will be limited to Vienna. The research could be done in every other city as well, as long as enough social media output is produced. Regarding the

fact that the Company DataScience Service GmbH¹ provides a validation model to match the beauty of neighborhood with real estate prices, Vienna is the obvious test case.

1.3 Papers Related to this Work

The following papers provide the basis of this study:

In 2010 Hochmair published the paper “Spatial Association of Geotagged Photos with Scenic Location”. In this paper the footprint of geotagged images along different routes was researched. The study reveals that Panoramio photos show a higher spatial association with user-posted routes when compared to fastest routes.

In 2013 Alivand and Hochmair published the study “Extracting Scenic Routes from VGI Data Source”. This study takes a closer look on which criteria has the biggest influence on scenic routes. In the setup VGI data from two platforms were collected for the area of California. In a next step different routes were compared in terms of their VGI output. Once again more scenic routes provided more VGI data.

Helbich et al. [2013] and Brunauer et al. [2013] collected data and developed an algorithm for the calculation of real estate prices in Austria. The algorithm includes indicators like infrastructure, education centers, the neighborhood and many more. Since the factor neighborhood is directly linked to the surrounding beauty of the area, this factor is regarded as the connection of the real estate model with the parameter observed in this study. The real estate model is provided by the DataScience Service GmbH and will be used as a validation of this study.

1.4 Approach

A Problem of previous studies [Hochmair 2010, Alivand & Hochmair 2013] was that the images had to be collected and classified manually. This prevents the case of the idea for large-scale applications. Thus, in this study, deep learning is used for the classification and the results shall answer the question, if this method is suitable for this kind of problem.

The goal of the research is not only to present a way how density heat maps could be generated out of VGI data, but also how to automate most of the process from gathering the data to the final model. Said process should be able to facilitate the update of the model when new data are available or the adaptation of the model to new regions.

This can be achieved with neural networks. The usage of a convolutional neural network is the state of the art in terms of image recognition. Therefore an own network will be build and trained in order to filter non valid data and bypass the task of manual classification.

¹DataScience Service GmbH is a DSS-Startup of the UT Vienna

The first step of the study will be how to collect the data and use it for setting up such a neural network. After the training a new dataset will be picked and classified. The filtered data will then be used to create the final model of a heat map. In a final step the heat map will be compared to a model of actual real estate prices to find out, if the accumulation of social media images is linked with housing prices.

2 Background

Chapter 2 focuses on the services, technologies and software which are used for the study. In the first part of chapter 2 two popular photo platforms are observed. These platforms will provide the data for the training and the final heat map. It will also be presented how data acquisition through this platforms work and which data could be obtained. The second part will give a brief look on deep learning and neural networks in particular. This technology will be used for data classification. After an introduction on deep learning a more detailed look will be given on the Inception V3, a trained neural network and how it can be used with the help of TensorFlow. The third and final section of chapter 2 will present QGIS, a geographic information system. A closer look on data types within QGIS is made and how they will impact the data of this study. The end of part three concerns coordinate systems within QGIS.

2.1 Photo platforms

2.1.1 Instagram

Instagram is a photo and video sharing platform. It is part of the Facebook branch and has about 700 million users. Every day thousands of images are uploaded to its servers. People use Instagram for sharing photos concerning nearly every topic one can think of. So, this app provides all information needed to create classification network. It provides a broad span of different topics and displays how different topics are presented on a social media scale. The idea is to use the images downloaded from Instagram to teach the neural network the variety of pictures, which are present on social media.

Images on Instagram are tagged with hashtags. These words associated with the image are referred to as hashtag since they have the “#”- symbol in front of the tag. Hashtags could simply describe objects which are in the images, persons who appear on those, shown locations or other associations made with the images. For example an image of the St. Stephan’s cathedral could be tagged with hashtags like: #St. Stephan’s cathedral, #Vienna, #1st district, #Stephan’s Platz, etc.

Like most of the image hosting platforms Instagram allows geotags for its images. Geotags are position hashtags. When an image is geotagged usually a longitude and latitude of the place where the image was taken is added. This position information varies in their precision. Some geotags are precise within a few centimeters some others within larger scale, for instance the geotag “Vienna”, which specifies the city as the relevant tag. This information could be accessed via Instagram Application Programming Interface (API). An API provides a user with application software building tools, the software for accessing and manipulating data on a specific platform.

Since Instagram has no particular focus on panorama images or images with a certain focus on the area, these images will only be used for training. Therefore, the Instagram API will not be used in this study but should be mentioned for the sake of completeness.

4K Stogram

This program is a tool for downloading Instagram files. The user is able to search by keywords, hashtags, and locations. Originally the tool is made for followers to download the images of people on Instagram to their local machines but could also be used to bulk download images concerning a specific hashtag or location.

4K Stogram makes bulk downloads of Instagram images very simple. Just by entering a specific hashtag or place every picture associated with it into the search bar, it will be searched and can be downloaded afterwards. Up to 3 different search terms can be used when the software is used as a freeware.

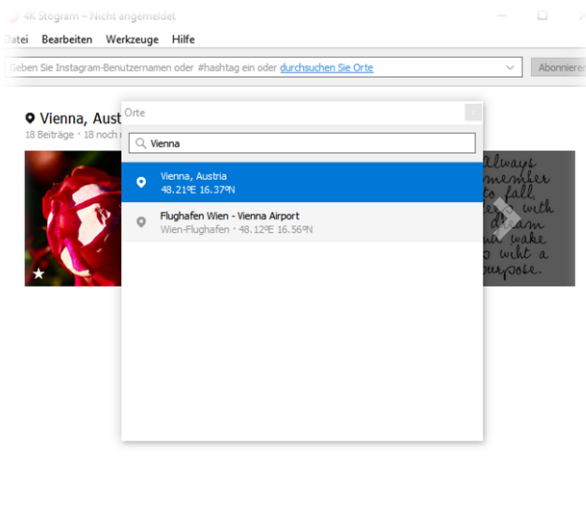


Illustration 1: 4KStogram search by location

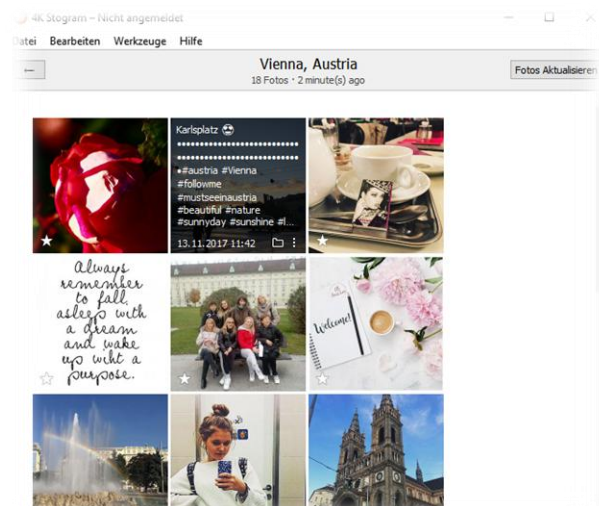


Illustration 2: 4KStogram data view

A downside of this software is that no metadata will be provided when downloading images. If the creator of the image is searched, a link to the user's Instagram side could be found in the image overlay within the program. Once the image is downloaded to the local hard disc, no further information other than the image itself will be available. The lack of metadata, especially location data is the reason Instagram images could not be used to create the heat map out of it. Therefore this program will only be used for gathering images for the training and test data, where no further information on the pictures will be needed.

2.1.2 Flickr

Flickr is an image hosting platform owned by Yahoo and has up to 70 million users. While Instagram function more as a temporary image sharing Flickr provides a well structured archive function. Instagram does images saving as well, but most of the time Instagram users

are rather interested in latest image not so much in older ones. Flickr on the other hand are used as storing platform. Flickr users usually upload image folder, with many images from the same topic, for instance a city trip. Therefore the saving of images is very well structured. This platform gives users the option how images are stored, if these are private or public. Like in Instagram, images get tags and could also get geotagged.

Therefore Flickr has a stronger focus on how images are taken and stores much more information of the image, like which camera was used, are there any special filters, and if there is GNSS data available in the form of an Exif-file.

Furthermore, a user could specify the license of an image. There are different license types, like creative commons, for commercial use, and some more. This fact comes in handy when the images are used for studies or other applications. With the information about the image's license the downstream user knows exactly what the image creator allows him to do with the image.

FlickrAPI

Like most of the popular sharing platforms Flickr provides an API as well. There are different download applications for Flickr as well, but, unfortunately, all of them have downsides. Every software, as far as this study can tell, is limiting the amount of images available for download to about 100. As the required data for this study should be a lot more than that, these applications are not useful.

By directly using the Flickr API these problems can be circumvented. The API enables the user to write his own code which fulfills the requirements. The Flickr API can be used with several different programming languages. In this study Python is used.

In order to use the API a key must be requested from Flickr. There are no other conditions for getting one than having an active Flickr account and write a short note for what it will be used. Then Flickr provides a "normal" and a "secret" key.

For creating a code with the Flickr API any text editor can be used. When data is requested via the API the user has access to all images which are public shared on Flickr. Since this is a huge amount of data, a pre filtering is necessary. For filtering the API provides the user with search parameters. The most important parameters are listed below. How they get used in this study will be presented in Chapter 4.4.

Important Search Parameters:

`ispublic`: If set on "1" -limit the search for pictures to public license

`media`: Since Flickr functions as a video and photo storing and sharing platform this parameter can be either set to "video" or "photo", if only one format is desired

bbox: When searching for geotags a bounding box can be used. This box functions as a search perimeter.

radius: As an alternative to the *bbox* a search radius can be used. When using this parameter, a center of the circle has to be passed to the code in the form of longitude and latitude.

accuracy: Allows to specify the accuracy of the geotag of an image

Specific hashtags could be used as search parameter as well. But since hashtags are very heterogeneous a lot of valid data could be lost and invalid data obtained.

2.1.2.1 Geotag

Flickr gives users the opportunity to geotag images with and without GNSS. This can be done on an OpenStreetMap (OSM)-Map. Geotags in Flickr are provided with different levels of precision. According to Alivand et al. [2013] only 3-4% of Flickr images are geotagged. Geotags will be categorized in how precise their position is. According to Hochmair [2010] precision 1 is world level, level 10 would be neighborhood level and precision 15-16 is street level.

Geotags are not equal to GNSS data and they are sensitive to errors. The geotag could have its origin in GNSS data or could be tagged manual. In the case of a manual tagging it does not necessarily mean that an image will be tagged in the correct spot.

2.1.2.2 GNSS

Some images in Flickr have real location data in the form of GNSS coordinates. When photo equipment with GNSS receivers is used, it guarantees that the picture was taken at the exact spot, except the Exif was manipulated. This is the case for 40% of the geo tagged images on Flickr.

2.1.3 Location-Metadata

The location metadata of images is important for this work. A differentiation has to be made in terms of this metadata. When working with social media images two types of geo locations can be found: Exif-files and geotags.

The exchangeable image file format (Exif) is a standard format which holds all the data which was created when the image was taken, like camera type, photo size, colors used and so on. It can also hold the GNSS information of an image, if it was added while taking the picture or appended afterwards.

Exif-files can only be acquired from Flickr when an image is captured in original size. Illustration 3 is taken by SandorSomkutiand². The description in Exif-format was accessed with an Exif-tools created by Phil Harvey. Next to the Exif tool form by Phil Harvey there are Exif reader and manipulation libraries for most programming languages.



Illustration 3: Kunsthistorisches Museum by Sandor Somkumi

The Exif-message of the image in Illustration 3 looks as follows:

```
ExifTool Version Number      : 10.37
File Name                    : 34370894934_43744dd4b6_o.jpg
Directory                   : C:/Users/
File Size                    : 9.0 MB
File Modification Date/Time  : 2017:08:31 10:39:27+02:00
File Access Date/Time       : 2017:08:31 10:39:20+02:00
File Creation Date/Time     : 2017:08:31 10:39:20+02:00
File Permissions             : rw-rw-rw-
File Type                    : JPEG
File Type Extension         : jpg
[...]
Make                        : NIKON CORPORATION
Camera Model Name           : NIKON D500
X Resolution                 : 600
Y Resolution                 : 600
Resolution Unit             : inches
[...]
Exif Version                 : 0230
Date/Time Original          : 2017:06:10 12:33:54
Create Date                  : 2017:06:10 12:33:54
[...]
Date Created                 : 2017:06:10 12:33:54.61
City                         : Wien
State                        : Wien
```

² Available at <https://www.flickr.com/photos/somkuti/34370894934/>

```

Country                : Österreich
[...]
Creator                : Somkuti
Rights                : Creative Commons Attribution-
ShareAlike 4.0 International License
Usage Terms           : Creative Commons Attribution-
ShareAlike 4.0 International License
Creator Work URL      : www.somkuti.at
[...]
Date/Time Created     : 2017:06:10 12:33:54
Digital Creation Date/Time : 2017:06:10 12:33:54
GPS Latitude          : 48 deg 12' 18.21" N
GPS Longitude         : 16 deg 21' 43.14" E
GPS Position          : 48 deg 12' 18.21" N, 16 deg 21'
43.14" E
Image Size            : 5095x3397
Megapixels            : 17.3
Scale Factor To 35 mm Equivalent: 1.5
Shutter Speed         : 1/640
Create Date           : 2017:06:10 12:33:54.61
Date/Time Original    : 2017:06:10 12:33:54.61
Thumbnail Image       : (Binary data 15479 bytes, use -b
option to extract)
Circle Of Confusion   : 0.020 mm
Field Of View         : 53.1 deg
Focal Length          : 24.0 mm (35 mm equivalent: 36.0 mm)
Hyperfocal Distance   : 5.13 m
Lens ID               : Sigma 17-70mm F2.8-4 DC Macro OS
HSM | C
Light Value           : 14.3

```

The Exif for this image was greatly reduced. Only some of the information of the file is presented, to get an idea how such a file looks like. The mandatory information which is needed for the model are the GPS latitude and longitude, the creation time and the image size. GPS Longitude and Latitude use the World Geographic System 84 (WGS 84).

The second type of location metadata is geotagged images. These images will get a geotag after the upload on the page. A concern regarding the geotagged pictures is the accuracy of the Geotag-location the images contains. Low accuracies of images, e.g. on city level, could create assembly points of images and therefore hotspots that are created only due to low accuracy.

Nevertheless, metadata is mandatory for this study. Without this information an image is useless in terms of the model since it could not be located and therefore would not be incorporated into the model.

2.2 Deep Learning & Neural Networks

Deep Learning is part of artificial intelligence and therefore part of information technologies. Since the beginnings in the 1950 this branch has made major developments. Especially in the recent decade big steps were made in order to train machines "intelligent" behavior.

[Schmidhuber 2015]

The idea behind deep learning is to recreate the structure of the biological neural systems and connect the single neurons like a brain would do. Based on these connections better handling of information takes places and the machine is able to learn. These learning models are called artificial neural networks. [Hijazi et al. 2015]

There are two main methods how to train a neural network:

- the unsupervised method, and
- the supervised method

In the unsupervised version of a neural network the network receives unlabeled training data [Mohri et al. 2012].The network should discover groups of similar instances within the data. Within this approach there is no a priori information about class labels or how many classes there will be. [Guerra et al. 2010]

For the supervised method, already classified/labeled data are used. The network will be trained on these already sorted data [Mohri et al. 2012].These sorted data cause benefits but also have disadvantages. The benefit of this method is that the network creates connections in a way it was originally intended. Therefore the network is able to make predictions about its accuracy. The downside is that sorting big data manually is very time consuming.

As stated above, in the recent years deep learning has made a lot of improvement. On the basis of this work different kind of networks were created. There are several types of networks, each of them being aligned for special tasks [Schmidhuber, 2015]. In this study a Convolutional Neural Network will be trained with the supervised training method.

2.2.1 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a feed-forward network and is used for deep learning. CNNs are the state of the art in terms of visual classification. [Szegedy et al. 2015]

CNNs are based on a significant design. It could be broken down into four major parts, namely the convolution layer, the pooling layers, non linear layers, and the fully connected layers. [Hijazi et al. 2015]

- *“The convolution layer extracts different features of the input. The first convolution layer extracts low-level features like edges, lines, and corners. Higher-level layers extract higher-level features.”* [Hijazi et al. 2015]
- A pooling or sub sampling layer reduces the resolution of the features and therefore makes the features more robust against noise and distortion.
- Non-linear layers are non-linear “trigger” functions to signal a distinct identification of likely features on each hidden layer.

- Fully connected layers mathematically sum a weighting of the previous layer of features, indicating the precise mix of “ingredients” to determine a specific target output result.

These four layer types could be used multiple times within one network. To get a rough idea how such a CNN could look like, the scheme of a CNN is presented in Illustration 4.

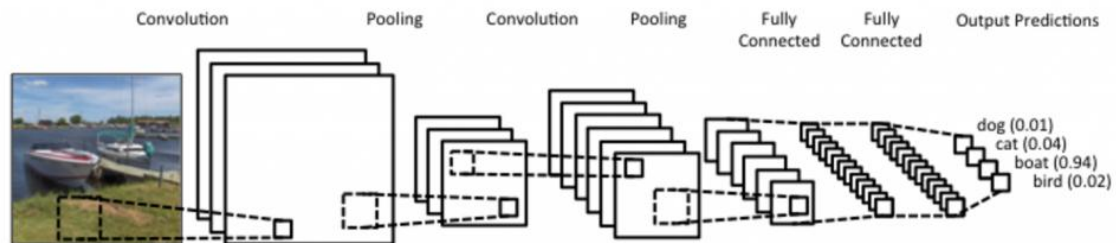


Illustration 4: Scheme of a CNN

The weights of a CNN function as shared weights. This means that the same filter is used for each receptive field in the layer. By doing this the memory footprint is reduced and performance will improve.

This is only a brief survey of a CNN. The structure of a CNN is much more complex and relies on many more parameters than the presented ones. For a more detailed introduction please refer to the scientific literature since this would exceed this study by far. For further readings on neural networks Schmidhuber (2015) provides a summary of the history and functionality of neural networks and additionally provides a summary of the most significant work in the field. For more detailed information on the newest and most important developments in CNN a list of the “9 Deep Learning Papers You Need to Know About” can be found on Adit Deshpande GitHub Blog on Understanding CNN’s at <https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>.

2.2.2 Inception V3

The Inception V3 a trained CNN created by Google. It won the ImageNet Large Scale Visual Challenge 2014 (ILSVRC14) and set the new standard for visual recognition. Every year since 2010 various scientists compete in this challenge to bring up the best CNN. The CNNs will be trained on the ImageNet data set and have to be able to classify images in 1000 categories. The CNN with the lowest error rate wins the challenge. Within this challenge the Inception V3 outperformed all other CNNs with an error rate of only 6.67%. Furthermore, the special design of the V3 uses fewer layers than other CNNs, which make it faster and more resistant to over fitting.

Overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably" [OxfordDictionaries.com]

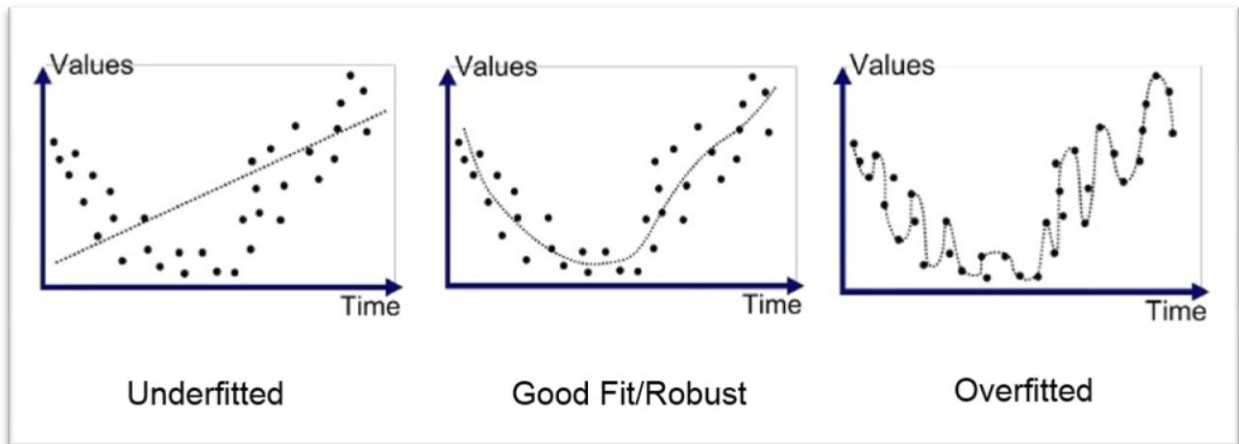


Illustration 5: Chart view of curve fitting

Illustration 5 shows how a curve can fit the data. An underfitted curve doesn't match the data at all and therefore will not be representative. In the case of the overfitted curve the curve matches the data too exactly. The problem with this in supervised CNN is that only the training data would fit for the classes. The data that are supposed to be classified would produce many errors since they will not be the same as the training data.

It is trained on over a million images and provides a solid basis for other networks. That high level of training could hardly be accomplished by a standard home PC. It would take weeks or even months to train a network with this data-rich input. The training was done by Google with supercomputers. Google allows everybody to work with their V3 CNN and build an own classification on top of it. The idea behind this is that the already very good trained network will function as core of a new network, which will be built around it. All the biases and neurons which are already adjusted will be maintained. Only the bottom layer of the network, also known as bottleneck, will be cut and replaced. Then the network will be trained on the new bottlenecks. The benefit of this technique is that a fully developed network could be used and therefore reaches a high accuracy, with much less data. This can be done with all kind of images, not only those, which are used in this study.

2.2.3 TensorFlow

TensorFlow is an open source library devolved by Google. It could be used for various numerical computations using data flow graphs. With TensorFlow different neural networks could be constructed. For this study a CNN is built with TensorFlow on top of the Inception V3 network. To accomplish this, the original V3 has to be modified.

Bottlenecks

A bottleneck is a term often used as the layer before the output. Since the V3 network is highly trained, most of the network will be untouched. Only the bottlenecks will be cut and replaced by own classes. This preserves the already trained layers of V3.

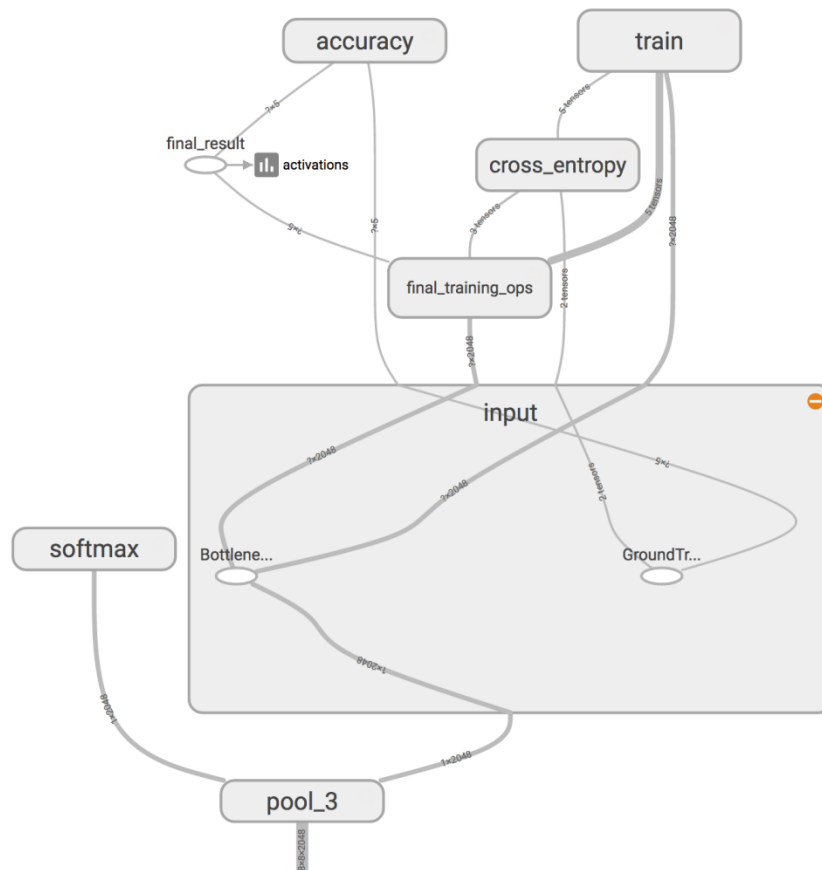


Illustration 6: Adding own classes

Illustration 6 shows a model of how the replacement of the bottlenecks takes place. After replacing the bottlenecks, the network will be retrained using every “new” image multiple times. Since the lower layers of the network remain as they used to be, their outputs can be reused.

In the training process each cached value for each image is fed into the bottleneck layer. The softmax node in Illustration 6 represents the output of the V3 model.

While training three important outputs are created by TensorFlow:

- The training accuracy, which shows the percentage of the images used in the current training batch that were labeled with the correct class
- The validation accuracy, which displays the precision on a randomly selected group of images from a different set.

- And the cross entropy, which is a loss function and indicates how well the training process is progressing.

These three are indicators on how well the network is trained and how precise it will be in terms of classification. For a good solid network, the training and validation accuracy should be preferably high and the cross entropy as low as possible.

If for example the training accuracy is high, but the validation accuracy remains low the network is over fitting. When a network is over fitting it memorizes particular features in the training images which do not support the classification process.

At the end of the training process TensorFlow provides an overall accuracy. The overall accuracy is determined by TensorFlow by self-testing the trained network with a random part of the training images. Therefore, the accuracy may vary in different runs. An overall accuracy in the range from 85%-99% is considered as a good value.

2.3 QGis

Quantum GIS or QGis is an open source geographic information system (GIS).

“A GIS is a system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data.” [Wikipedia on QGis]

QGis allows managing data in different layers. Each layer can hold one type of data and various metadata. Two types of data are supported by QGis - raster and vector data. Raster data are represented by the minimum number of grid cells containing it and vector data by its geometry: point, line, or polygon. Both data types will be used for this study. After the data are classified from the CNN it will be saved as point data and therefore as vector data. This data will serve as the basis for the heat map. The heat map itself is a raster data set.

2.3.1 Vector data

Vector data is defined by its geometry. For this study the classified data will result in a file where each image is stored with its GPS coordinates. This file will then be converted in vector data, so that each image will be represented by a point on the map.

Since the model should be created for Vienna, also the city borders are required. This data will be represented as polygons, also vector data. It could be done with polylines as well, but most of the computations used in this study are much easier to handle with polygons.

2.3.2 Raster data

Raster data are defined by their pixels. This type of data is often used to show information beyond geometric objects, for example to show rainfall trends or fire risks over a region.

The final product of this paper will be a heat map. This kind of map displays the point density over a certain area and assigns colors depending on how high the density value is. The consequence of this map will be to find point accumulations.

2.3.3 Coordinate System

QGIS provides its users with different coordinate systems. There are many different local and global systems. Local coordinate systems have the advantage of not distorting angles and areas as much as global systems for a specific area. In Austria for example the MGI³ Austria Gauß Krüger System is used as official system. A disadvantage of a local system is that it only works for this specific area. Using a local system from Austria for another country would have a tremendous impact on all the displayed features and would be simply very wrong.

To overcome problems like this, global systems come into play. This system could be used for the whole earth and earth close space. One of these global systems is the World Geodetic System 84 or WGS 84.

Since GNSS data are stored in the WGS 84 and the data used in this study mostly relies on GNSS metadata the whole model will be built in this system. An additional advantage of preferring WGS 84 over a local system like the MGI Austria GK system is that the whole process stays transferable for use on a different study area.

³Transl.: Military Geographic Institute or Militärgeographisches Institut

3 Methodology

In this section of the study the methods and techniques of the research are presented. Furthermore, a detailed description will be given of decisions being made and why. Part one gives an overview of the hardware and operation system (OS) being used. The used set-up is not mandatory and many other set-ups would work fine as well. The second part introduces on how valid and non-valid images could be categorized. Note that the data sources used in this paper are not exclusive. The reason for using Instagram respectively Flickr images are explained in Chapter 2.1.1 and 2.1.2. Alternatively, every other social media photo sharing application can be used for training the network as long as location data are available. The conditions of being social media data and the availability of the location data are important. In advance of the study different approaches were discussed like using Google image search or real estate images. However, these two approaches would not be expedient to achieve valid results. First of all, they are more prone to error when used as training data. Simply because a Google image search as well as real estate images would never create the same variety as social media. And secondly the distribution of the images, if location data are available, could be uniform and controllable based on the search terms used. Within part three geotag locations will be contrasted with GNSS locations. It will give a look into advantages and disadvantages of those locations types. Next a closer view is given on the neural network being used for this study. It makes clear why a CNN is the matter of choice and what advantages TensorFlow provides. The final section takes a look into the output format of the CNN and the software used to prepare and analyze the data and finally to create the heat map.

3.1 Hardware and OS:

For this study an MSI GX660R notebook is used. It runs with an Intel Core i7-740QM CPU and 8 GB DDRIII RAM. VGA support is possible and highly recommended for training neural networks, but since this machine only got an ATI Radeon GPU it is not applicable in this case. GPU processing is only supported for NVIDIA GPU's.

With regard to the operation system Linux Ubuntu (OS Geo Live 10.0) based on Ubuntu 16.04 Trusty is used. The choice for a Linux based system rests on the fact that most of the literature and documentation is Linux based as well. Therefore, it appears to be reasonable to use such a system for better troubleshooting.

3.2 Valid images

Since the data for the model will be collected from social media platforms, it is very important to use only pictures, which actually make sense in the context of an area. In this part the presorting of images is described. Since the CNN is trained under supervision, the training classes set-up is done manually. In the next passage the different images types will be explained and why they are valid or not.

When image data are obtained by social media a lot of different image types can be found. These images can be roughly categorized in 9 different groups. The categorization was developed during the classification process. This categorization is subjective and might be seen from another viewpoint, especially when put in a different context. The nine classes are:

- Event: This class contains all pictures taken at events and/or clubs. The class takes no account, if the event happens outdoor or indoor, both will be put into this class
- Food: As the name suggests, this class contains all kinds of foods and drinks.
- Indoor: Every image taken inside will be stored here, except for indoor event images.
- Meme: All files which show any kind of writing, commercials or painted images will be part of this class.
- Photo-shoot: This class contains photo shoots of persons with no real link to their surroundings. They could be taken outdoor and indoor.
- Random: All the pictures which do not fit in any of the other classes will get in here.
- Selfie: Self-portraits which are not relevant to the area.
- Stuff: Any kind of buyable objects get in here. Animals, too, fall into this class, since it was unnecessary to create another single class for them.
- Valid: This is the most important class. It holds all the images which are valid for the final model. All images with a link to the area.

Every class of images is classified as invalid with the exception of the valid class. The aim behind every class is to sort images which are alike. This is the particular reason for putting indoor events into the event class rather than into the indoor class. The object is to collect images with similar color schemes in a class. Event images often display bright and colorful lights. Even if the image is taken inside, these components will characterize the image. Therefore, by putting an inside event image into the event class it is possible to preserve the color scheme of both the event and the indoor class.

3.2.1 Invalid classes

3.2.1.1 Event

Event is, like most of the class headers, widely understood. The event class contains all images which are taken in clubs, on concerts or on public events which take place in Vienna, like the Christkindlmarkt or the Donauinselfest. Since event photos are very popular and events produce a large number of images, these will be filtered out, since they would distort the image distribution.

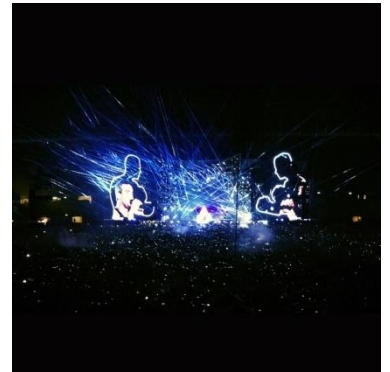
Examples:



Example 1



Example 2



Example 3

In the above examples different images of this category could be seen. Example 1 shows how an outdoor event could look like, in this case the Christkindlmarkt at the Rathausplatz. Under normal conditions this image would be valid, but since it was made during an event it will be categorized as invalid.

Example 2 shows an indoor event. The previously mentioned color scheme with the colorful light could be seen clearly.

The third example is from a live concert. Especially at the end of June a massive amount of these images is produced in the course of the Donauinselfest. Since this abrupt increase of social media output is directly linked to the festival and not to the beauty of their surroundings it has to be filtered.

Most of these images share a similar color scheme, since many event photos have a lot of artificial, most of the time colorful light displayed on them.

3.2.1.2 Food

Eatable or drinkable goods make up a lot of images uploaded to social media platforms. Since most of these images have no connection to their surroundings at all they will be classified as invalid.

Examples:



Example 4



Example 5



Example 6

Although some of these images show parts of the area around them, the main focus of the pictures is on the consumable good. It can be speculated that these images can be taken anywhere.

There are some food images which are exceptions of this speculation. Some images of food are made with a direct link to the surroundings. These images make up a small part of food images. An example for this will be presented in Chapter 3.2.2.

3.2.1.3 Indoor

For indoor images the case is pretty simple. Since they are taken inside there is no correlation to the area outdoor.

Examples:



Example 7



Example 8



Example 9

In the case of indoor images three main types of image content could be encountered. In Example 7 we can see the first type – the private setting. Other examples for these settings are family gatherings, groups of people who meet for dinner or just displaying the inside of their home.

The second type is public places. Since Vienna offers a lot of architectural objects, a lot of these images can be found. They show both classical and modern objects. But like all images in this category they have no value in terms of the areal beauty.

The third type is simple displaying commercial content. This type will be not encountered as much as the other two, but has to be mentioned for the sake of completion.

3.2.1.4 Meme

The category meme is a collection of different image contexts. It contains all sorts of artificially created images, screenshots and images with added text in them. Also, pieces of art will be placed in this category. The header for this group is nonjudgmental, but since 90% of images are not photos but graphically created objects, what would nowadays be described with the neologism meme, they were lending this name to the category.

Examples:



Example 10



Example 11



Example 12

The three example images show files which fall into this category. Example 10 shows a classic meme. In this case it is a motivational meme. Since meme are most of the time pretty similar in their appearance only one example is shown although they make up most of the category.

Example 11 and Example 12 show creative products. Since these are not very common they are merged in the meme category instead of creating an extra category.

3.2.1.5 Photo-shoot

The category “photo-shoot” includes all images with a focus on persons but no actual selfies. Since a lot of these pictures are from professional photo sessions they lead to the category's name.

Some images in this class are really challenging in their classification. In other classes it is much clearer if an image is valid or not, but since a lot of the images in this category are taken outside and might even display the surrounding area, it takes some afford to decide whether they have a focus on the area or not.

Examples:



Example 13



Example 14



Example 15

The examples shown above are three clear cases for non-valid photo-shoot images. None of these images have another focus than the person itself.

In the section on the valid category an example will be shown, which would belong to this class, but in contrast to the images displayed above it will have a focus on the area next to the person and therefore count as valid.

3.2.1.6 Random

The random category is a relatively open class. All images which could not be classified in another class will be collected in this one. The context of these images is difficult to summarize. A lot of the pictures show tattoos, finger nail styles or even some artistic photographs.

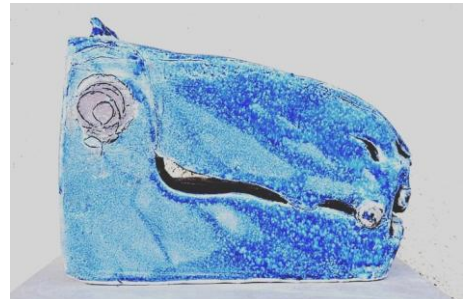
Examples:



Example 16



Example 17



Example 18

Example 16 to Example 18 show some of the image context which appears more often in this category. Since the category is more or less an open one, images with a totally different context could be found here as well. Some of the images might show clouds or geometric structures.

3.2.1.7 Selfie

In terms of a selfie, the assumption is made that most of the selfies are taken with no direct correspondence to the area. These pictures have to be filtered out. Otherwise they would falsify the model. Part of the validation process is to distinguish between selfies with reference to the surroundings and those focused on a person only.

Selfies make up the second group of images which could be very challenging some time. An example for valid selfies will also be displayed in the valid section.

Examples:



Example 19



Example 20



Example 21

Like in the examples above we can see the three major types of selfies - the close selfie, the mirror selfie and the group selfie. The close and the group selfie types are able to produce valid images. Nowadays it becomes very popular to make selfies in front of touristic buildings. So, if one of these two types is taken outside with only a little focus on the person/s itself it could count as a valid image.

The mirror selfie is never able to create a valid form of context in the image. Most of the images are taken inside and will be invalid for this fact alone.

There is the possibility that the outdoor area is reflected within the mirror of a selfie. For these images it has to be decided individually if they are valid or not. It should be decided based on whether the creator had the intention of showing the area or the area just happened to be in the reflection.

3.2.1.8 Stuff

The Stuff category is for all kinds of buyable goods with the exception of food. Also, objects, like traffic signs and lights and animals will belong to this category.

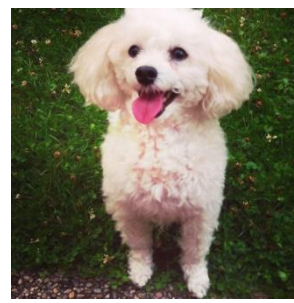
Examples:



Example 22



Example 23



Example 24

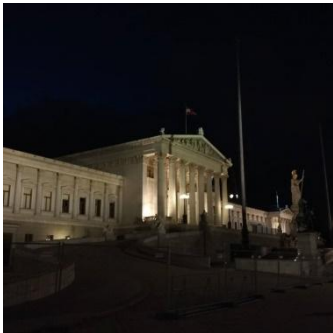
Most images found in this category will simply display buyable goods like cloths, accessories or cars. Relatively often traffic lights will be encountered. In 2015, on the occasion of the Euro Vision Song Contest and the Life Ball traffic lights showing couples were installed in Vienna⁴. These couples are still a quite popular subject.

Other than buyable goods and traffic signs also animals can be found in this category. Since they make up a rather small quantity of images an own category would be unnecessary.

3.2.2 Valid class

The Valid class holds all images with a focus on the area. For some images it is quite clear that they belong to this class, like the images shown in Example 25 to Example 27. The whole purpose of these images is to display the area or certain pieces of architecture. Most of the images in this class show a similar context.

Examples:



Example 25



Example 26



Example 27

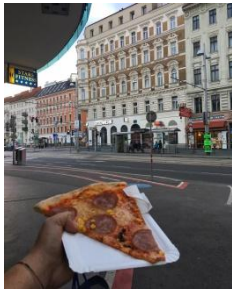
Images in this class must not show architecture at all. Also, panoramic images in the form of parks or vineyards are valid as well, as long as it can be assumed that the intention of the photographer was to display the surrounding area.

As described in the previous chapter there are also images which are classified as valid images although an object or creation type of other classes is within the image.

The next examples show an image of some invalid class which will be classified as valid, because the focus is not only on the object or person as such but on the surrounding area as well. The only excluded categories will be the event, the indoor and the random classes. The event class since it will always falsify the distribution, the indoor class since the outdoor beauty of an area will be studied and the random class since this classification is not able to produce valid images.

⁴ Original: "Wiener Ampelpärchen"

Examples:



Example 28: Valid Food



Example 29: Valid Meme



Example 30: Valid Photo-shoot



Example 31: Valid Selfie



Example 32: Valid stuff

All of the examples presented above have in common that they could be classified in an invalid class, but all of them share a focus on the area as well.

Since this classification is done manually it might be hard sometimes to decide if an image belongs to an invalid class or not. The guidelines should always be: what was the creator's intention to display in the image and could the same image be taken somewhere else.

To make things clearer take a look at Example 31. The intention of the selfie was to take a picture with the Schönbrunn Palace in the background. The main purpose was to display the Castle and so the image couldn't be taken somewhere else.

If the intention of the creator is not clear, most of the time it is better to decide in favor of the invalid class to preserve the integrity of the valid class.

3.3 GNSS VS Geotag

When it comes to VGI data, especially when dealing with social media data, studies tend to use geotags. Like the papers presented in Chapter 1.3 Alivand & Hochmair [2013] and Hochmair [2010] use geotags for the location of the image along different routes. Using geotag is quite common, but not always the best approach for locating data. In a study on the accuracy of Flickr's Geotag data by Claudia Hauff [2013] it was found that geotags in Flickr could quite differ in location from where they actually were taken. In her study she also pointed out that when pictures were taken away from tourist hotspots, especially areas with low Flickr output, they tend to be even worse in terms of accuracy. She researched the accuracy in different tourist hotspots in Europe. One of those was St. Stephen's Cathedral in Vienna. St. Stephen's Cathedral is one of the major hotspots in Vienna with several million

visitors every year. This location produces a mass of images on social media. And even on this huge image producing spot the pictures show deviations of several meters. One must keep in mind that Flickr produces geotags on different levels of accuracy. But even if only the highest level will be used, deviations will occur. This might not be a problem when working on projects with small scale maps, but when it comes to larger scales, high accuracy is mandatory for the result. Since in the route studies the scale which was used was quite small in comparison to this study, the accuracy of geotag is adequate.

Another phenomenon which occurs with geotag is the assembly point. Some social media platforms tend to create assembly points in hotspot areas.

For example: In Vienna there is an event called “Sand in the City” (SitC). It takes place on the property of the Vienna’s ice skating club⁵ from May to September. During SitC various food and cocktail bars are placed on the 6,000 m² big area. When tagging an image in Facebook with a hotspot, e.g. SitC, the image will not be tagged at the exact position where the image was taken but rather on a point centralized in the location. So, when afterwards the location is searched all images taken there will be located in the same spot, even though they were made in different parts of the area.

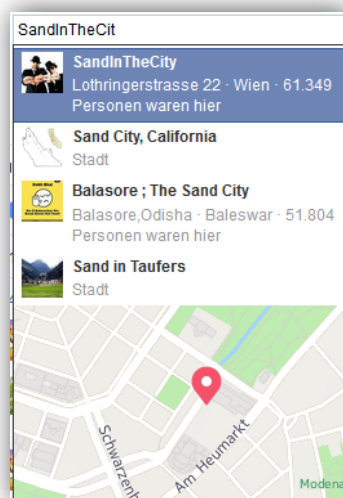


Illustration 7: Geotag Assembly Point Facebook

As mentioned above, the accuracy of the VGI needed for this study is much higher. Luckily in the recent years the use of cameras with GNSS-chips increased. This factor will be exploited for the research done in this paper.

As described in Chapter 2.1.3 actual GNSS data will be stored in an Exif-file with the image. Since the accuracy of this data could be trusted more than geotag, only images with an active GNSS location will be taken into account. Therefore, it can be stated that the images

⁵ Original: Wiener Eislaufverein

which are used are much more reliable and more precise in terms of their geo location rather than using geotags. It should be mentioned that the correctness of the GNSS location data of the images used for the heat map are not checked. Since manipulation of Exif-files can be ruled out with a high degree of certainty, it is assumed that the location data will be correct.

Another advantage of this approach is that the location displayed in an image corresponds to its geo-location. Geotags will be often added afterwards, when the image data is uploaded. Therefore, errors regarding the correct location could occur. The manipulation of an Exif would be possible, but requires some programming skills and consequently is very unlikely to happen.

3.4 CNN & TensorFlow

Data classification is a slow process when done manually. This is one of the main reasons why machine learning techniques come in handy. Machines are efficient when performing simple tasks over long periods because they follow scripts and are not affected by human deficits like a limited attention span.

In the past different approaches for classifying data were created. Approaches using different types of decision tree structures were successful, but in terms of visual data classifications a more complex technique was needed. This is when neural networks or deep learning comes into play.

CNNs have shown in the past that with their help computers, who cannot visually interpret are efficient in finding patterns, are able to classify images with high accuracy. Neural networks are increasingly used in scientific research and record a success in various fields. This is the simple reason why a CNN is also used in this study. The classification could be done with other techniques as well, but none of these could encounter the issue with such an efficiency and accuracy like a CNN. Especially when the Inception V3 network was released for everyone it opened a whole new way of image classification.

TensorFlow provides the link between programming languages and the V3 network. With the help of TensorFlow and the V3 network everyone is able to train and use a high end neural network which would occupy a whole team for several months to build. Of course, it would be possible to build a CNN from the scratch, but it would take a lot of computing power and a lot of time and would very likely not be as accurate as the retrained V3. It simply would not be cost efficient.

3.5 Data output & GIS

There are several different ways to store the data output of the CNN. After the classification process the image itself is not needed anymore. Only some of the metadata and the geo-location need to be passed on for the analyses. To do so the output will be saved in a csv-

file. A csv-file comes with some advantages rather than storing the output right into a database. The created file would be very low in terms of storage capacity. Even if a large amount of images is classified a csv-file will hardly exceed a couple of MB. If new data is classified simply a new csv file could be created and merged afterwards with a database management system or GIS.

The classification process itself requires a lot of temporary memory. Writing in a csv-file is much more cost efficient in terms of RAM usage compared to constantly updating a database. Furthermore, most GIS software and database systems are able to read csv-files.

For the analyses of the data QGis is used. The analysis could be done with any GIS software available. QGis was chosen since it provides a heat map plug-in for the final model. Furthermore it was already used in the geoinformatic research group of UT Vienna and is well known.

4 Workflow

The result of this study should be a heat map, which shows the distribution of images of Vienna published in social media. Areas with a high density of images should be considered as more beautiful. The major task to accomplish is to automatize the whole workflow as far as possible, so follow-up research can be done just with little to no changes.

The first part of Chapter 4 will show how to set up a training data set. It will describe how the data is obtained and show the derivation of the classes presented in Chapter 3.2. Afterwards the network will be trained, important commands for the training explained and finally the best set up for the CNN presented. The follow up section will show how the data for the model is gathered and what conditions the downloading script has to fulfill. This part will result in a collection of images. In part four the images will be classified and the important data metadata will be extracted. The output after the classification will be a csv-file ready to feed into a GIS. The final section explains how the raw point data gets prepared for further computation and how a heat map is generated.

4.1 Creating a training set

When working with neural nets, especially convolutional networks, quite some training data is needed. The network has to learn the difference between valid and invalid output. Like seen in Chapter 3.2 social media produce a sheer amount of different images types. To be prepared in the best possible way a lot of different pictures will be used.

For the training set Instagram images are used. The software “4K Stogram” is used for downloading the data. This program makes bulk downloads of Instagram images very simple. Just by entering a specific hashtag or place, every picture associated with it will be searched and can be downloaded afterwards. For the training data simply the location “Vienna” is used. 4k Stogram will provide more than enough data concerning this location.

About 11,000 pictures were downloaded. It is anticipated that about 10 percent of these data will be unusable for the training, because of format problems or simply just because the data are videos. This leaves about 10,000 images for the training and the test data. The number of images used for training could be lower and much higher as well.

The reason for this quantity of images used is efficiency. Since manual classification is time consuming and the expected data set for the final model will be about 20,000 to 40,000 images in size, 10,000 images provide a solid basis for training of the network.

In a next step the training classes must be created. This is the most work-intensive part, because it has to be done manually. Nine classes will be designed, like they were presented in Chapter 3.2. After the classification of the training set is done, from each class 5-10% of the images will be separated and held back for testing the network afterwards.

After the classes are set up the derivation of the images looks as follows:

Table 1: Image Derivation Instagram

Image Derivation Instagram				
Class	Images	Test set	Total	Percentage
Event	120	15	135	1.33
Food	600	42	642	6.34
Indoor	1500	88	1588	15.68
Meme	750	51	801	7.91
Photo-shoot	270	44	314	3.10
Random	400	46	446	4.40
Selfie	1500	113	1613	15.93
Stuff	1100	105	1205	11.90
Valid	3100	284	3384	33.41
				100.00
Total	9340	788	10128	
Original Downloaded			11000	
Useless Data			7.93	

Another approach of populating the sets could be done by searching images with the precise hashtag the class should contain. For example, in terms of the class selfies, the term “selfie” is searched by 4k Stogram. This method would be ineffective for two reasons. The first downside is that even if a specific term is searched the results will be contaminated with non-fitting images for this class. So, the process of validation must be accomplished as well.

The second downside lies in the fact that the aim for each training set should be to gain the ability to deal with all different images produced by social media. If the collection of the training data is done by isolated search terms, the possibility that the neural network encounters images, which were never seen during the training, is much higher than when the training is done by mixed images, in terms of search.

But this method could help fill classes, which hold too few images for training. In the original 9-class setup this method would have come into play for populating the event class, but since finally only a valid and non valid class for classification was used (Chapter 4.2), there was no need to use this method for the study.

The number of images for each class varies greatly, since not all topics are presented equally strong on Instagram.

4.2 Training of the Neural Network

TensorFlow runs on different OS. For this Study Linux OS with a Docker container is used. Google provides installation guides for all major OS and also offers different builds of TensorFlow. It can also run outside of a container as well. Since the documentation for image classification is very detailed for a Docker container, this build up is used.

The whole set up for TensorFlow and Docker can be found on the following pages:

- TensorFlow: <https://www.tensorflow.org/install/>
- TensorFlow for poets: <https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/#0>
- Docker: <https://www.docker.com/community-edition>

The presented TensorFlow set up uses Version 0.12.1.

First the Inception V3 network will be needed and must be obtained and retrained. This will be accomplished with the following Docker command:

```
# In Docker
python tensorflow/examples/image_retraining/retrain.py \
--bottleneck_dir=/tf_files/bottlenecks \
--how_many_training_steps10000 \
--model_dir=/tf_files/inception \
--output_graph=/tf_files/retrained_graph.pb \
--output_labels=/tf_files/retrained_labels.txt \
--image_dir /tf_files/classes
```

TensorFlow loads the retrain.py script. In the next step the bottlenecks are cut and the new bottlenecks are created. The script can be set to use a predefined number of iteration steps. In this case the number of iterations is defined as 10,000. By default the script uses 4,000 iterations.

In the following three command lines the script gets the information where to store the new model, the output graph and the labels are created. The labels will have the same labels as the classes which are used for retraining.

The last line of code refers to the location where to find the training classes. Note that the Docker container will be linked to a folder while installing. Therefore, the code only needs the directory within the already linked folder.

Depending on the configuration of the computer being used, the iteration steps and the number of training images used it may take 20 min to several hours to train the new model. For this build it took about 2 hours. But we have to keep in mind that there are about 10,000 images used and more iterations than the default value.

The iteration steps are the number of reruns of the own image classes TensorFlow makes for retraining the network. After this the new bottlenecks of the network are created.

At the end of the training a message will be displayed which shows if the model is completely trained and how accurate it will be. The accuracy of the model will be calculated on the factors described in Chapter 2.2.3.

For a first test of the network a training set up as follows was used:

Table 2: First test set up

Class	Images	Total	Percentage
Indoor	288	288	16.19
meme	170	170	9.56
selfie	287	287	16.13
Stuff	339	339	19.06
Valid	695	695	39.07
			100.00
Total	1779	1779	

The first test set hits an accuracy of 91.4% with 5 classes. This set has a pretty good accuracy although five classes have been used. This is because the classes the model was trained on were relatively small. The main class valid had about 700 images and each of the other classes only included about 300 images. Therefore, the spread within a class is not as broad as it would be with about three times respectively ten times the size.

For the final model the network was trained on all images on several different class set ups, to find the best configuration. Starting with the original attempt of 9 classes it turns out, that the network hits a low accuracy of about 74%. Since the network has too many different decisions it could choose from, the percentage for an error is larger than if there are fewer classes.

The results of the different set ups can be seen in the table below:

Table 3: Configurations of the training set ups and their accuracy

Number of Classes	Iteration	Accuracy
9	4000	72.00%
9	8000	74.00%
5	4000	83.40%
4	4000	86.20%
3	4000	88.30%
2	4000	92.90%
2	8000	93.00%
2	10000	93.50%

Since the accuracy with two classes is much higher than with more, two classes will be used for the final model. In a next step the iterations steps are raised. With 10,000 iterations an accuracy of 93,5% was hit. As can be seen in Table 3 the increase of the iteration from 4,000 to 10,000 has very little impact on the accuracy of the CNN – only 0.6%. The more important fact was to reduce the number of classes. Also note that the accuracy will vary each time the CNN is trained, even with the same images, since the training images of the network are randomly chosen from the training sets.

The final image Distribution for the training data looks as follows:

Table 4: Image distribution of the final set up

Class	Images	Test set	Total	Percentage
Non-valid	6240	504	6744	66,59
Valid	3100	284	3384	33,41
				100,00
Total	9340	788	10128	
Original Downloaded			11000	
Useless Data			7,93	

The picture distributions for the other set ups can be found in the attachment.

4.3 Testing the model

Although TensorFlow test the network itself, a supervised controlled test will be done after the training. TensorFlow provides the classification script. Some changes have to be done in order to test the new model. In the original script only a single image could be classified. For simple image classification this would be enough, but since a lot of image classifications must be done, the original script must be manipulated. So the original script will be changed in order to accept folders as input, and runs through the files within a loop. This is done with a pretty standard listdir command:

```
work_class = sys.argv[1]

work_dir = os.path.join("/tf_files/", work_class)
...
for files in os.listdir(work_dir):
    iffiles.endswith("jpg"):
        image_path = os.path.join(work_dir, files)
```

Very important is to move the graph_def section out of the loop.

graph_def:

```
withtf.gfile.FastGFile("/tf_files/retrained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
```

```
_ = tf.import_graph_def(graph_def, name='')
```

Otherwise the script would reload the graph with every loop and reach his RAM buffer limit of 2 GB quickly. In the original script this is not important since only a single image without a loop is tested. The graph_def is the part of the script which loads the previously trained classification graph. When moving the graph_def out of the loop the performance of the script is much better.

In the next lines of code the actual images classification takes place. For this the original code section is inherited.

```
# Read in the image_data
image_data = tf.gfile.FastGFile(image_path, 'rb').read()
# Loads label file, strips off carriage return
label_lines = [line.rstrip() for line
intf.gfile.GFile("/tf_files/retrained_labels.txt")]
with tf.Session() as sess:
    # Feed the image_data as input to the graph and get first
    prediction
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')
    predictions = sess.run(softmax_tensor, \
        {'DecodeJpeg/contents:0': image_data})

    # Sort to show labels of first prediction in order of
    confidence
    top_k = predictions[0].argsort() [-len(predictions[0]):] [::-1]
```

It is not necessary to know all values of each class for a tested image. The interesting fact for testing is only if the images reach a high score in the class in which they are already classified manually. So every class from the test data will be tested separate. For speeding up the process, the output statement will be changed too.

In order to only get the values of interest two conditions are set up:

```
for node_id in top_k:
    human_string = label_lines[node_id]
    if human_string == "non valid":
        score = predictions[0][node_id]
        if score > 0.9:
```

The code example is showing the test of a non-valid folder. By changing the human string to "valid" the same conditions can be used for a valid folder.

For the classification score 90% (0.9 in the code) of the tested class was selected. After several runs the result show that the reached score in a certain class is pretty high most of the time. Especially when testing the valid class, all images which reach a lower score than 0.9 are actual non valid.

Another small change is done with the output statement. If a classification score hits lower than 90% in the class it was manually classified to, the file name will be stored in a CSV-file together with its class scores.


```
f = open(image_path, 'rb')
tags = exifread.process_file(f)
out_csv.write("%s\n" % (image_path))
```

If there is an error, the problem file could be looked into and it can be checked if there was an error with manual classification in the first place or the network reacts in a wrong way.

4.4 Data for the model

For the creation of the final model new images are needed. Instagram images could be used for this as well, but since this platform has its focus on several contexts another source of images must be obtained. The two major distributors for panorama image data are Panoramio and Flickr. Panoramio would provide more image data with geo-location than Flickr. Unfortunately, the project Panoramio ended in November 2016, so Flickr images have to be used.

There is different software for downloading Flickr images, similar to 4k Stogram. Unfortunately, since the procedural part of geotagged images in Flickr is so low, a lot of useless data would be acquired. By using bulk download software a very large amount of storage space would be needed.

In order to circumvent the problem an own script is created. There are some important demands to the script to overcome storage capacity problems.

- Firstly, only pictures with an Exif and a valid GNSS entry should be acquired.
- Secondly, in order to reduce storage space it should be avoided to download the image in original size. It should be tried to get hold of the smaller sizes of the same image since images in original size have up to 20 MB these days.
- Thirdly, a strategy is needed to obtain the Exif, even if the image does not have its original size.
- Fourthly, only images in the area of Vienna should be acquired.

If these four demands are met, the storage space as well as the downloading time are reduced drastically.

Downloading script

Since the images which will be downloaded are searched and downloaded from Flickr the FlickrAPI is used. At the beginning of the script both API keys (Chapter 2.1.2) are needed, in order to get a valid site request from Flickr.

Example key⁶:

```
flickrAPIKey = "380681cf13e54df65df45465df42e4bf584" # API key
flickrSecret = "eac554d5f4d32f07 " # shared "secret"
```

⁶These are made up combinations, only for the sake of showing how these keys look like.

Next a time window will be created. This means that the images will be filtered in order to get only images in a specific period of time. Flickr saves images with an upload timestamp in the form of GNSS time. This is a time code in seconds, starting on January 6th, 1980. From then on it counts up till present time. For the downloading the image time window will be limited to every image between 30.06.2016 to 30.06.2017. The window setup was chosen to cover a whole year in order to get images from all seasons.

```
mintime = 1467244800 # 30.06.2016
endtime = 1498780800 # 30.06.2017
```

For the next step blocks of said time window must be created. Downloading the complete time window within one go would cause some troubles. It is very likely that Flickr itself cancels the download by blocking the request key because the request is simply too big. Another downside of using the time window in the form of one block would be that, if the script has some sort of error, it would be very hard to find out where to start again without looking through all the images twice. To circumvent the problem the script is partitioned in shorter periods of time. The script will always summarize the images of one week into one block. When one block is downloaded the next block will start with the beginning time of the last block's ending time and so on.

```
timeskip = 604800 #one_week
maxtime = mintime+timeskip
```

Now a first filter is added to the script, but this is a very vague one, since it only limits images to a certain time. In Chapter 2.1.2 FlickrAPI four search parameters were introduced. Four of those will now help to put a more precise filter into the script.

- The “ispublic” parameter set on one ensures that only images with an open license form will be acquired.
- The “media” parameter limits the data to only photos, since videos have no value for this study.
- With “bbox” a geographical search window will be created. The bounding box will cover more than just Vienna, to make sure that also the outer districts will be included.
- The “accuracy” parameter is not need since only GNSS metadata will be obtained. If there is a need for geotag data, maybe for getting more data, this parameter must be set to its highest level of 16, when working on a scale like this study does.

Further, two new parameters will be introduced and included- the parameter “page” and “per_page”. These two are no actual filter parameters. They will be included to get a better segmentation of the image blocks. Since one block could get very large in terms of image number, it strikes as a benefit to subdivide them for a better overview.

One block of a one week time window in this setup could hold up to 3 000 images. When the script encounters an error and stops, it will be an advantage to know more precisely where to start over. Therefore, the block will be subdivided in pages of 250 images each.

```
rsp = fapi.photos.search(api_key=flickrAPIKey,
    ispublic="1",
    media="photos",
    per_page="250",
    page="1",
    bbox="16, 47, 17, 49",
```

...

The image request with the parameters will be done twice. At first for just gathering the information if images are available for this time block and a second time in the actual downloading part of the script.

Only images with an active GNSS entry in their Exif are of interest. So, when the photo is requested the script takes a look if an Exif is available and if it is accessible. When the user denies the access of the Exif file, the script will simple get no valid response to the Exif request.

```
try:
    photo_exif = fapi.photos.getExif(api_key=flickrAPIKey,
    photo_id=photo_id)
except:
    pass
```

If the request is successful, the code looks for an Exif entry in the form of GPS latitude. It could be done with longitude as well. To check for one of both entries is enough, since if there is any location entry at all, it will hold both.

When the tags are available the information is pulled and formatted. The reason for changing the format is important for the creation of a new Exif file. More explanations will follow in the specific part.

```
if exif:
    avail_tags = []

    for tags in exif:
        avail_tags.append(tags['tag'])

        if "GPSLatitude" in avail_tags:
            for tags in exif:
                tag = tags['tag']

    if tag == "GPSLongitude":

        lon_str= tags['raw']['_content']
        elem = lon_str.split(" ")

        lon_deg = int(elem[0])
        lon_min = int(elem[2].replace("'", ""))
```

```
lon_sec = int(float(elem[3].replace("'",
""))*100)
```

In the code example above only the extraction and format transformation for the GPS longitude is shown. The same applies to the GPS latitude.

The reason for extracting the GPS data is to bypass the need for the original file. With the action of collecting the necessary metadata upfront any image size could be downloaded. Afterwards a new Exif file will be created. So, a massive reduction in storage capacity can be achieved.

Now the images have to be downloaded. Flickr offers various sizes of each image. Since most of the original images are quite large in size, smaller sizes are desirable. Original images could have up to 20 MB and large images might have a resolution from 2048*1152 up to 5 MB. Most of the time images with a high resolution like 2048*1... have a second size with a resolution in the format of 1048*582 available. So the third largest size will be acquired when the original size is available and the second largest when only large is available.

Since there is no conformity for image sizes the result may vary between medium and large. But this is perfectly fine for the requirements of this study. Both sizes have enough pixels for the CNN to classify on.

```
for sizes in avail_sizes:
    size_labels.append(sizes['label'])

if "Original" in size_labels:
    size = avail_sizes[-2]

    source = size['source']

    img_name = source.split("/")[-1]
    img_path = os.path.join(img_dir, img_name)
```

With this lines of code the average image size will be reduced to about 0.5 MB.

In the two bottommost lines the image will be saved in a specific directory. Now the script checks if an image is downloaded. If the image is not downloaded the script simply passes this file and continues with the next file in line.

```
im = Image.open(img_path)

gps_ifd = {
    piexif.GPSIFD.GPSLatitude: ((lat_deg,1), (lat_min,1), (lat_sec,100)),
    piexif.GPSIFD.GPSLongitude: ((lon_deg,1), (lon_min,1),
    (lon_sec,100)),
    piexif.GPSIFD.GPSDateStamp: img_time
};

exif_dict = {"GPS":gps_ifd}
exif_bytes = piexif.dump(exif_dict)
```

```
im.save(img_path, exif=exif_bytes)

f = open(img_path)
tags = exifread.process_file(f)
f.close()
```

In the final part of the script the new Exif file is created. Three kinds of information are needed here. The first two are the abovementioned GPS latitude and longitude. The third one is the image capture time.

For writing the Exif file the piexif library is used. Piexif allows for manipulating, deleting or creating Exif data. When creating data, especially GNSS coordinates, piexif demands a certain format, since all Exif files are in conformity in their information formats. Like mentioned in the section about extracting the GNSS information the format has to be observed. GNSS location data will be read out as one string. For writing the Exif degree, minute and second integers and float for second's format is demanded. In order to meet the demands the string is split and transformed to integer respectively float.

For some computation the image capture time could be for interest. To collect also this information the image capture time will also be saved. In contrary to the GNSS location the image time does not have its origin in the Exif but rather comes from Flickr metadata. There is also a time value for the image capture time in the Exif called GPSTimeStamp. This would be with certainty more accurate than Flickr metadata. But the work in progress has shown that too few images actually hold this value. Since time is only an additional factor the accuracy of Flickr is more than enough.

4.5 Classify images

For the final model the downloaded images have to be classified and the required information has to be stored. Since the image itself is only needed for the validation of an image, there is no need to save the images again. If an image is classified as valid only some information must be saved – the location, the capture time, its image path and for additional monitoring its validation score.

The location data is needed to create a point where the image was taken in the QGis later on. Like mentioned before the time value is not necessary but a nice feature to have. Image path and validation score are stored for safety reasons. If there is an unusual clustering of points or geometrical arrays, which seems very unlikely to happen by pure coincidence, these specific images can be reviewed in their correctness. The validation score serves as some kind of second safety net for the classification process. When in reviewing it turns out that an image was classified as valid but is not, the CNN has to be revised.

Most of the script used for the final classification can be inherited from the test script. There are only some changes to the output statement. In the test script image path and score of

wrongly classified images were saved. Now the images which are classified correctly will be saved in the csv file with the previously discussed additional information.

In the final build only images with a score of 90% in the valid class will be accepted. If an image satisfies these conditions its metadata will be saved.

```
fornode_id in top_k:
    human_string = label_lines[node_id]

    if human_string == "valid":
        score = predictions[0][node_id]
        if score > 0.9:
            print('%s (score = %.5f)' % (human_string, score))
            f = open(image_path, 'rb')
            tags = exifread.process_file(f)

            if "GPS GPSLatitude" in tags and "GPS GPSLongitude" in
            tags:
                lat_list = tags["GPS GPSLatitude"].values

                lat_deg = int(str(lat_list[0]))
                lat_min = int(str(lat_list[1]))
                lat_sec = str(lat_list[2]).split("/")
                lat_sec = float(lat_sec[0]) / float(lat_sec[-
                1])
                lat = lat_deg + lat_min / 60. + lat_sec /
                3600.

                lon_list = tags["GPS GPSLongitude"].values
                lon_deg = int(str(lon_list[0]))
                lon_min = int(str(lon_list[1]))
                lon_sec = str(lon_list[2]).split("/")
                lon_sec = float(lon_sec[0]) / float(lon_sec[-
                1])
                lon = lon_deg + lon_min / 60. + lon_sec /
                3600.

                time = tags["GPS GPSDate"].values

                out_csv.write("%.6f;%.6f;%s;%s;%.2f\n" % (lat,
                lon, image_path, time, score))
```

Once again the coordinates have to be formatted to an actual number. Like discussed with regard to the downloading script (Chapter 4.4) the output format type of the Exif file is a string.

4.6 Creating the Heat map

For analyses and manipulation of the data and creation of the heat map QGIS Version 2.14.3 – Essen is used.

4.6.1 Load the data in QGIS

For the creation of the heat maps QGIS will be used. Since the output of the classification script is a csv file the data could be implemented as follows:

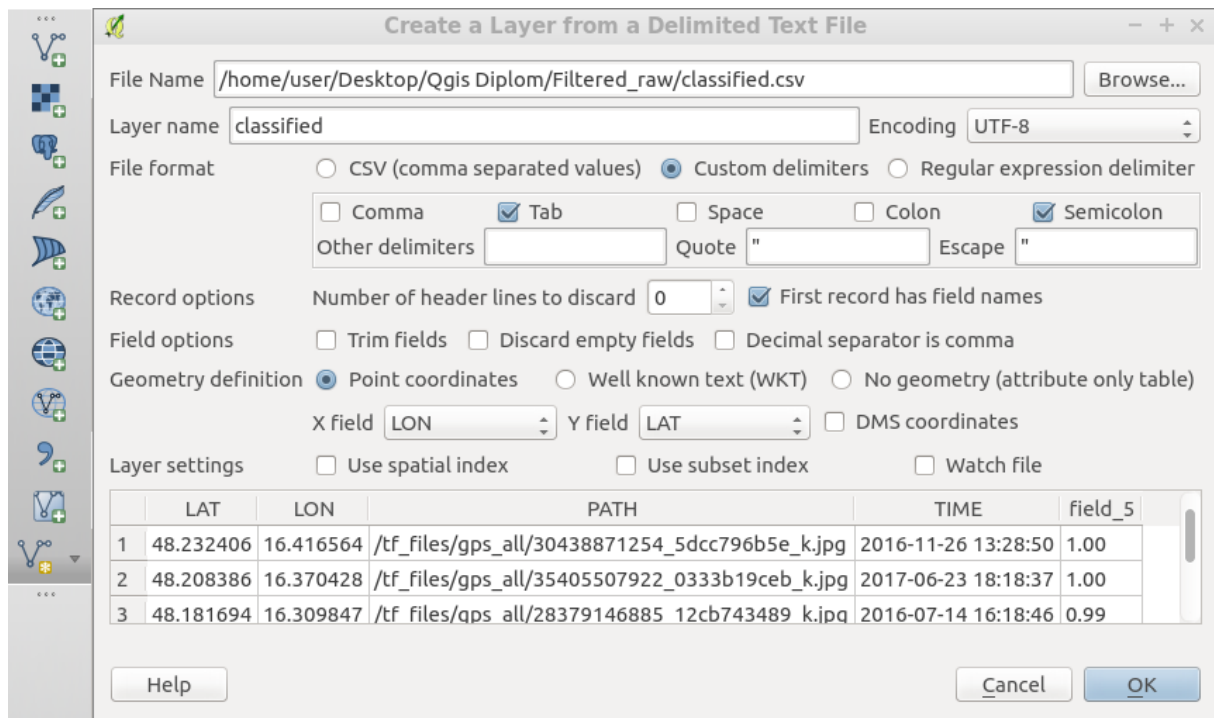


Illustration 8: Creating layer form text file; QGIS

The created csv-file with the classified images is added with “create a layer from a delimited text file”. The columns in the csv are separated with a semicolon. The X field is assigned to the longitude column and the Y filed to the latitude.

In the next step the coordinate system has to be assigned. Since all of the point data is from GNSS devices they are in the WGS 84 system and the same system is chosen for the QGIS project. If the output is needed in a local system like MGI GK projection for Austria it can be transformed with QGIS.

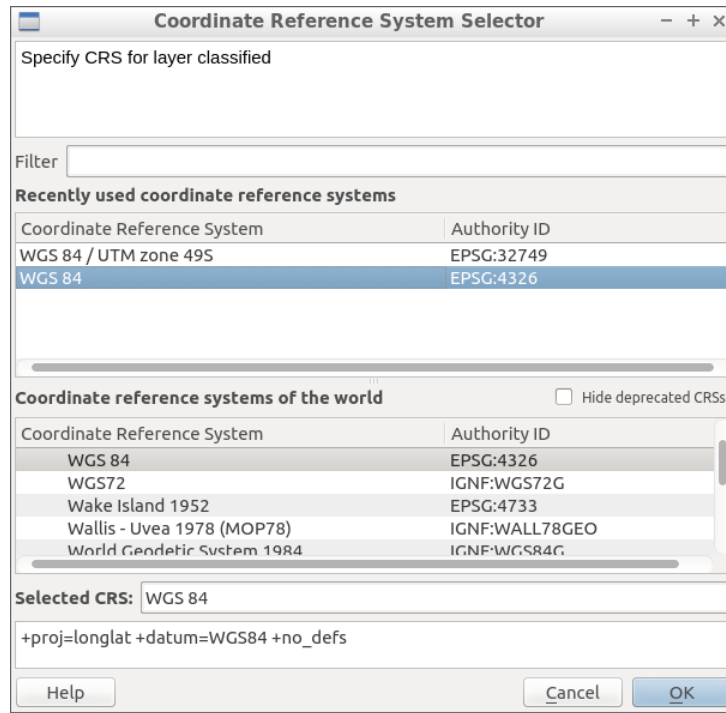


Illustration 9: Coordinate System Selector; QGIS

Now the data is available as point data in QGIS. Since the bounding which was used for a first filter in the download script was generously chosen, all points out of the boundaries of Vienna will be cut out. This also applies to points which did not get correct coordinates during the format transition.

For doing this a data set with the boundaries of Vienna is needed. This can be obtained from the vienna.gv.at website.

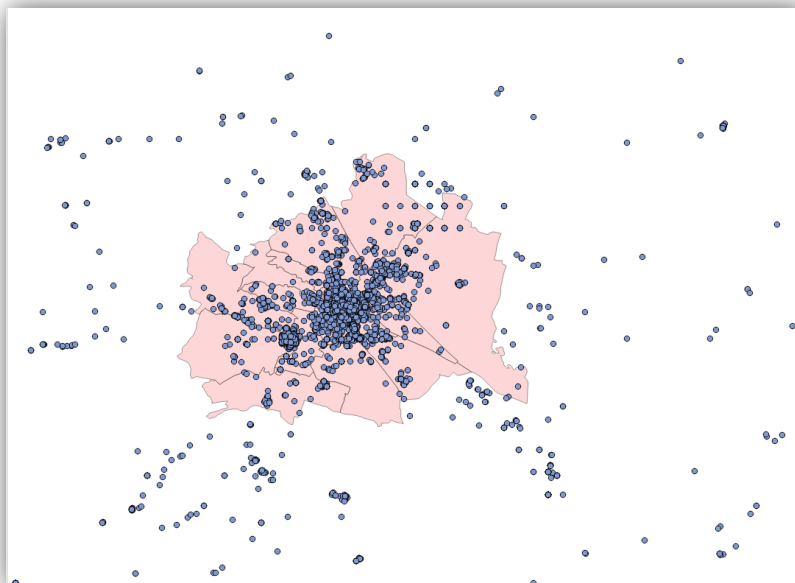


Illustration 10: Raw data and boundaries of Vienna; QGIS

4.6.2 Preparation of the data

For the filtering process the “Select by location” tool in QGIS is used.

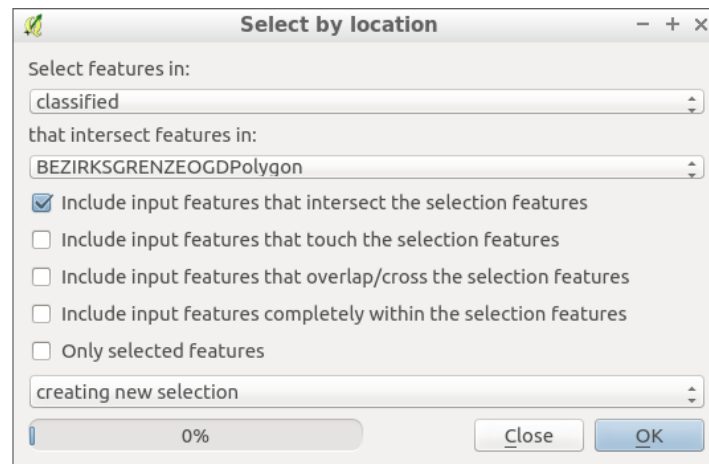


Illustration 11: Select by location tool; QGIS

As seen in Illustration 11 the tool needs to know which layers are compared. As a first feature the classified points are selected and the boundaries layer serves as an intersect feature ⁷. It is very important to create a new selection for this, since the raw data should not be changed as a safety measure.

The result shows all points within the boundaries of Vienna. The resulting layer will be named “Core”.

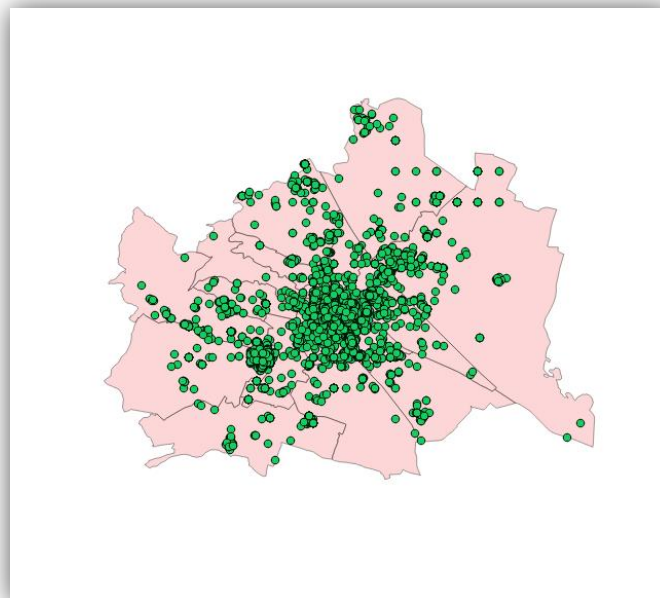


Illustration 12: Data points within Vienna; QGIS

⁷BEZIRKSGRENZEODPolygon

4.6.3 Heat map tool

For the creation of the heat map the heat map plug-in of QGIS is used.

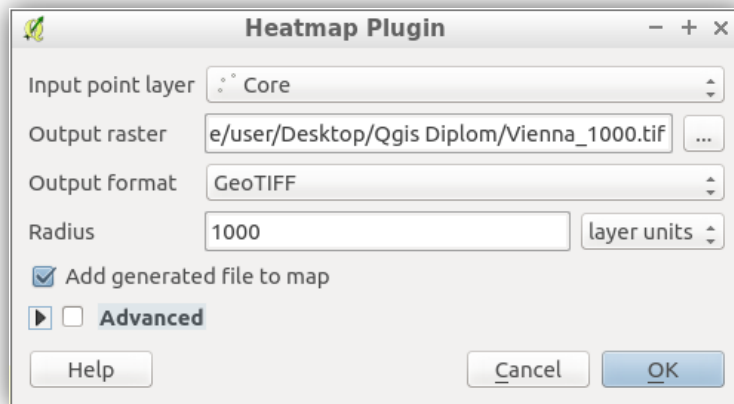


Illustration 13: Heat map Plug-in; QGIS

As input layer the core layer is chosen. The output format will be a GeoTIFF⁸. The radius indicates the radial influence of each point.

In the next step the now created heat map needs to be prepared in order to get a better visual form of the map.

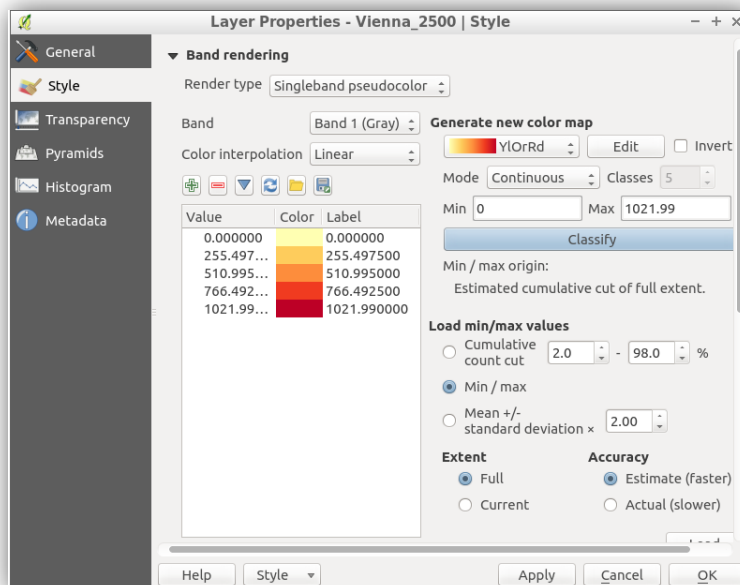


Illustration 14: Layer preparation; QGIS

⁸GeoTIFF is a public domain metadata standard which allows georeferencing information to be embedded within a TIFF file.

Firstly, in the style tap the render type will be changed from “Singleband gray” to “Singlebandpseudocolor”. Next a color palette is chosen- in this case yellow to red (YlOrRd). Now the Min/max box in the “Load min/max values” section needs to be selected and confirmed with the “Classify” button. In the window next to settings the newly created classes will be shown. By approving the selection of the classes with the “OK” button on the bottom of the tool window, the heat map will be changed and will appear in the new colors.

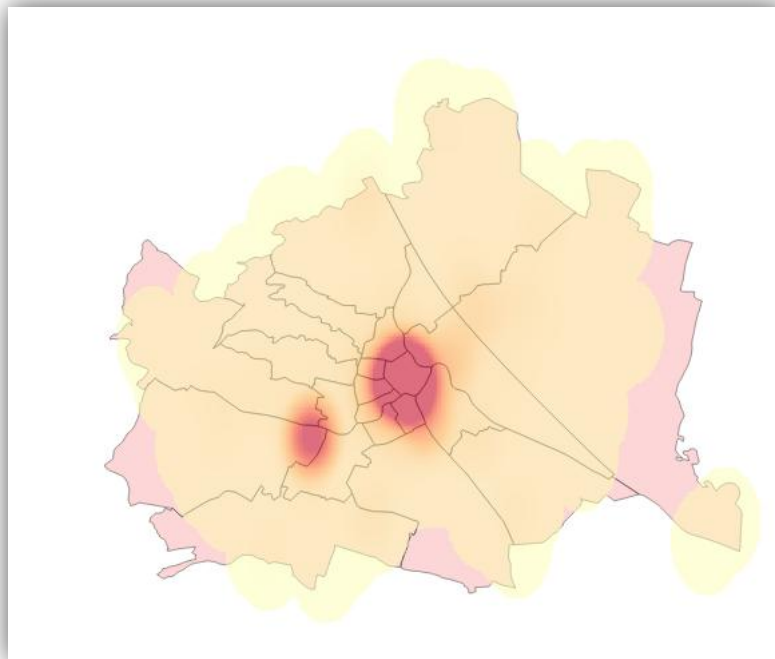


Illustration 15: Finished Heat map; QGIS

4.7 Challenges

In the process from the beginning to the final heat map some challenging situations will occur. In this part of the study some of the major challenges will be listed in order to help future work based on the presented techniques to avoid them. The order of the challenges is based on the time they took place along the road of this work.

4.7.1 Testing the model

This mathematical view of images is the biggest challenge for visual deep learning. A machine is good in finding patterns or fast computations, but lacks in competence when it comes to the interpretation of visual input. We have to keep in mind that a CNN is the actual attempt to teach a machine to make decisions based on the context of a visual image.

In Chapter 2.2.1 the functionality of a neural network is described. Each image, which will be classified, will be broken down to numerical matrices. The network itself does not look at the subject an image really displays, but rather at the numerical composition of the colors and where they are set.

So, when the neural network classifies images it relies on the numbers a pixel has. This way of interpreting images can cause some rare errors. Images with a totally different context will be classified as valid and vice versa.

In Illustration 16 a hand with painted fingernails can be seen, a quite common motif for today's social media. Images like this are normally classified in the "random" class and therefore as non-valid. Several images which display the same content of painted fingernails are in the training set for the non-valid class, but nonetheless this image will be classified as a valid one.

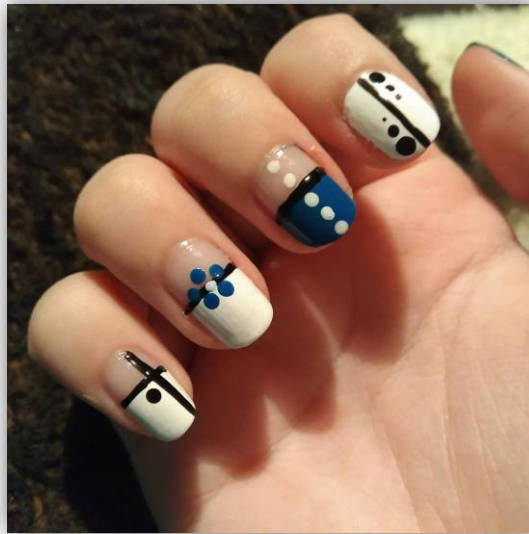


Illustration 16: Image falsely classified as valid

Errors like this one can occur from time to time. A clearly non-valid image will be classified as valid with a score from over 90%. Since this problem only represents a disappearing small quantity it will be not significant to the final result, but has to be mentioned.

4.7.2 Connectivity issues

During the download of the images connectivity problems can occur. One of the main reasons for this concerns connection problems with the Flickr server. The consequence is the abort of the ongoing downloading season. Unfortunately, there is no way to make sure this problem does not occur at all. Time and time again the connection could fail on either of the client or server sides. Putting all trust in one script could be very challenging. As a counter measure it is possible to run multiple scripts at the same time. Flickr allows up to three parallel site/server requests. Three scripts with shifted time windows will run in parallel. With this the overall downloading time can be cut to a third.

5 Results

In this chapter the results of the study and the final model will be presented. The first section of this chapter presents the image statistics from the downloaded content. Furthermore, it will give detailed statistics with regard to the distribution of cleared data in single districts.

In section two the final model will be presented. After that a closer look at the hot spots with a high image density is made. The overall model will be compared to the single district models for a better understanding of the distribution of images in these hot spots.

5.1 Image Statistics

5.1.1 Downloaded data

When using the downloading script presented in Chapter 4.4 a total amount of 272 pages for the time window of one year was found on Flickr. A page will hold 250 images. This makes up a total of around 68 000 images for the chosen bounding box. These are the images that have a geotag. According to Alivand et al. [2013] this is the case for about 3-4 % of all uploaded images.

68 000 images were further analyzed with the downloading script. From these about 40.5% have a valid GPS entry in their Exif file. This leaves us with 27 562 images for classification. Compared to the total images created in one year and uploaded to a social media platform this is only a small margin. But we have to keep in mind that these images are already heavily filtered.

5.1.2 Classified images

In the next step the images were classified. 27 562 images passed the first level of filtering and were fed into the CNN. After the classification a total of 10 109 images were classified as valid. This is nearly 37% of the downloaded data. When we look back to Chapter 4.1 it can be seen that this value seems to be realistic. In the manual classification of training data about one third of the images were valid.

5.1.3 Point data for the model

Since the bounding box for the data was generously chosen some of the classified images will be cut away simple because they are outside of the boundaries of Vienna (Chapter 4.6.2).

This leaves a total of 8 348 images for the creation of the heat map.

Table 5 gives an overview of the quantity of data that was viewed and used for the heat map.

Table 5: Summary of viewed image data

	Total Pages	Total Images	Images GPS	Filtered by CNN	Located in Vienna
	272	68000	27562	10109	8348
Percentage Total		100.00%	40.53%	14.87%	12.28%
Percentage GPS			100.00%	36.68%	30.29%
Percentage CNN				100.00%	82.58%

Table 6 shows the distribution of the points within Vienna of the prepared data. More than 40% of the point data are located in the first district, followed by 14.4% in the thirteenth district and 9.5% in the third. About 60% of the whole data is located in the 1st, the 3rd and the 13th district which barely cover 12% of the area of Vienna. The first district alone claims 40% of the data, while representing about 0.7% of the total area. This is properly due the fact that these districts are very high frequented by large amounts of tourist.

The consequence of this is that a lot of the area is poorly covered. Taking a look in the “Images per km²” column it can be seen that more than half of the districts have less than 10 images per square kilometer.

Especially districts which have a larger area have this problem. Smaller district logically need fewer images to get a good coverage. When comparing the 6th and the 22th districts it can be seen that the 22th district has nearly double the amount of images in total numbers. But compared to the size of the district the 22th has about 3 images per km² whilst the 6th has about 120 images per km².

However even a high image count per km² does not necessarily mean that the whole district has a good coverage. When a district has hot spots which create the majority of the images within this district it might be that the rest of the area is poorly covered.

Table 6: Distribution of images per district

District	Images	Percentage	Area m ²	Area km ²	Images per km ²
1010	3575	42.82%	2868772.83	2.87	1246.18
1020	433	5.19%	19241987.03	19.24	22.50
1030	794	9.51%	7397707.75	7.40	107.33
1040	174	2.08%	1775207.97	1.78	98.02
1050	126	1.51%	2011638.71	2.01	62.64
1060	175	2.10%	1455085.32	1.46	120.27
1070	281	3.37%	1608247.40	1.61	174.72
1080	59	0.71%	1089945.72	1.09	54.13
1090	190	2.28%	2967385.67	2.97	64.03
1100	91	1.09%	31828479.59	31.83	2.86
1110	93	1.11%	23256370.01	23.26	4.00
1120	72	0.86%	8102975.21	8.10	8.89
1130	1203	14.41%	37714743.43	37.71	31.90
1140	140	1.68%	33762716.42	33.76	4.15
1150	26	0.31%	3918422.39	3.92	6.64
1160	67	0.80%	8672846.22	8.67	7.73
1170	23	0.28%	11391352.69	11.39	2.02
1180	37	0.44%	6346852.23	6.35	5.83
1190	168	2.01%	24944167.42	24.94	6.74
1200	43	0.52%	5710483.17	5.71	7.53
1210	125	1.50%	44443215.69	44.44	2.81
1220	337	4.04%	102299426.02	102.30	3.29
1230	116	1.39%	32061794.96	32.06	3.62
Total	8348	100.00%	414869823.85		

Table 7 shows the statistical characteristics of the images can be seen. Because of the images clustering in some districts the ranges of the minimum and maximum counts of images in a single district diverge strongly. Also the 0.75-quantile reveals that for 75% of the districts the image count is below the mean.

Table 7: Statistical characteristics of the points

Mean	362.96
0.25 Quantile	69.50
0.75 Quantile	235.50
Sum	8348.00
Min	23.00
Max	3575.00
Range	3552.00
Median	126.00

5.2 QGIS

Illustration 17 shows the final model. In the model each point has an influence radius of 1000m. For a better orientation an OSM-base map and the district boundaries were added to the heat map layer.

At the first sight three hot spots can be detected. The first one in the 1st district becomes pretty obvious after viewing Table 6. Since a lot of points cluster in a small area the heat map value is very high. Two more hot spots can be detected in the 3rd and 13th district. The hot spot in the 3rd district covers the area around the Belvedere Garden. The hot spot in the 13th district and lies in the area of Schönbrunn. In comparison to the 1st district these two hot spots do not have the same value for the whole area. For a better visibility the three major hot spots were enlarged and can be seen in Illustration 18 for the 1st district, Illustration 20 for the 3rd, and Illustration 22 for the 13th.

It should be noted as well that the classes which can be seen in Illustration 17 have the same class width. Therefore the first class, starting from zero to 377, could contain areas with a good coverage. But since the values in the three major hot spots are very high the areas with fewer points but good coverage get lost.

Since the radius for the final model was chosen to be 1000m it makes it very difficult to say how the points are distributed within the district, especially when looking at the 1st district.

In order to get a better idea of the distribution of points within the hot spots for each of the spots a second heat map only of the area of the hot spot was created. These heat maps have smaller influence radii than the overall map of Vienna. It is also important to mention that these heat maps are created only of the points within their area. A direct comparison of these additional heat maps with the final model or with each other is therefore not permitted.

The values within the single district heat maps are completely different from the overall map. The reason for this is simply that by reducing the influence radius the touching area of the points is reduced as well and therefore the values decrease. This creates a local view of the points and it is easier to determine where they are really clustering.

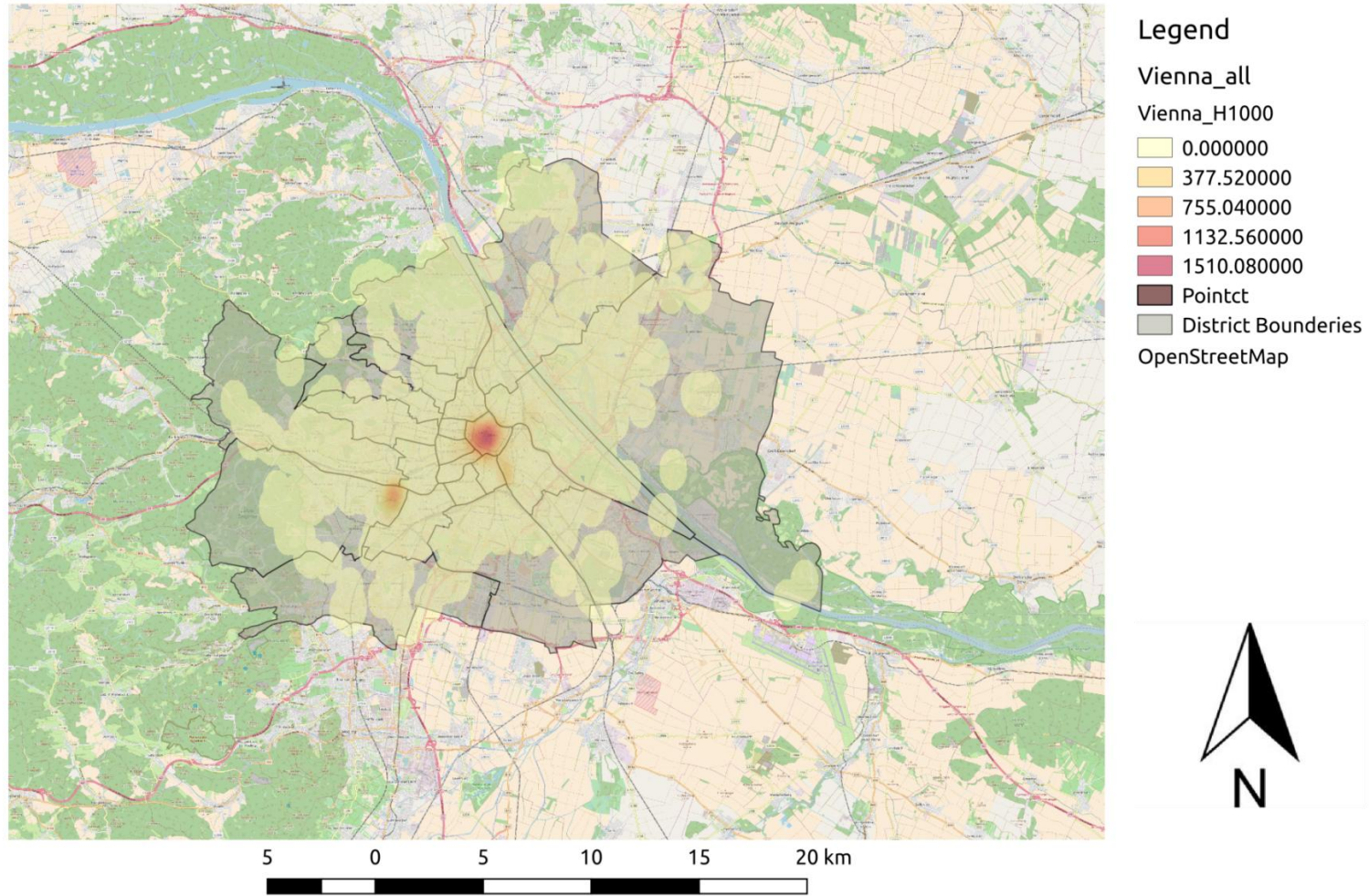


Illustration 17: Final Model for Vienna

5.2.1 1st district “InnereStadt”

Illustration 18 shows the enlarged hot spot of the final model. The value for the whole area is about 246 images. Since the 1st district is relatively small and also includes the largest number of points, this high value is obvious.

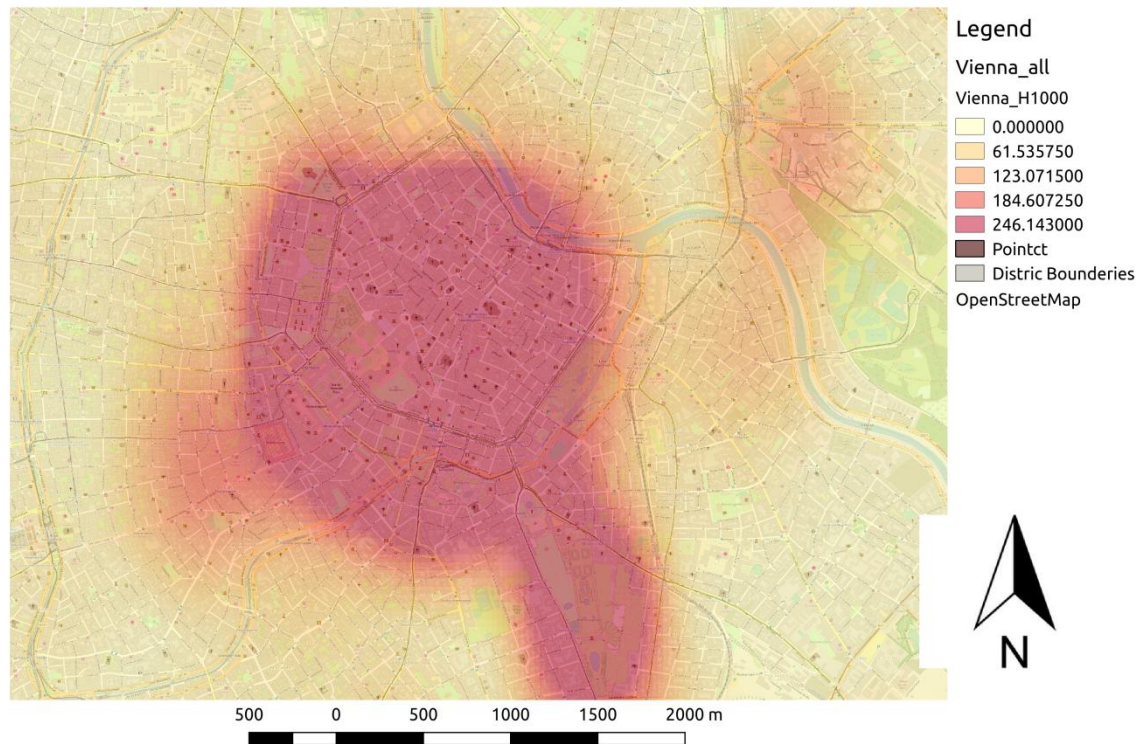


Illustration 18: Enlarged hotspot 1st district

In Illustration 19 the heat map for the single district is displayed. The radius for a point was reduced to 100m. Now local hot spots can be seen, like the St. Stephan’s Cathedral in the center or a little far west the “Michaelerplatz” in front of the “Hofburg”.

What all these local hot spots have in common is that they are at or nearby of popular tourist locations in Vienna. But it is very difficult to generalize this for the whole city. The first district is Vienna’s old town and there is an attraction behind every corner. Therefore to derive a generalized statement out of it could lead to a wrong assumption.

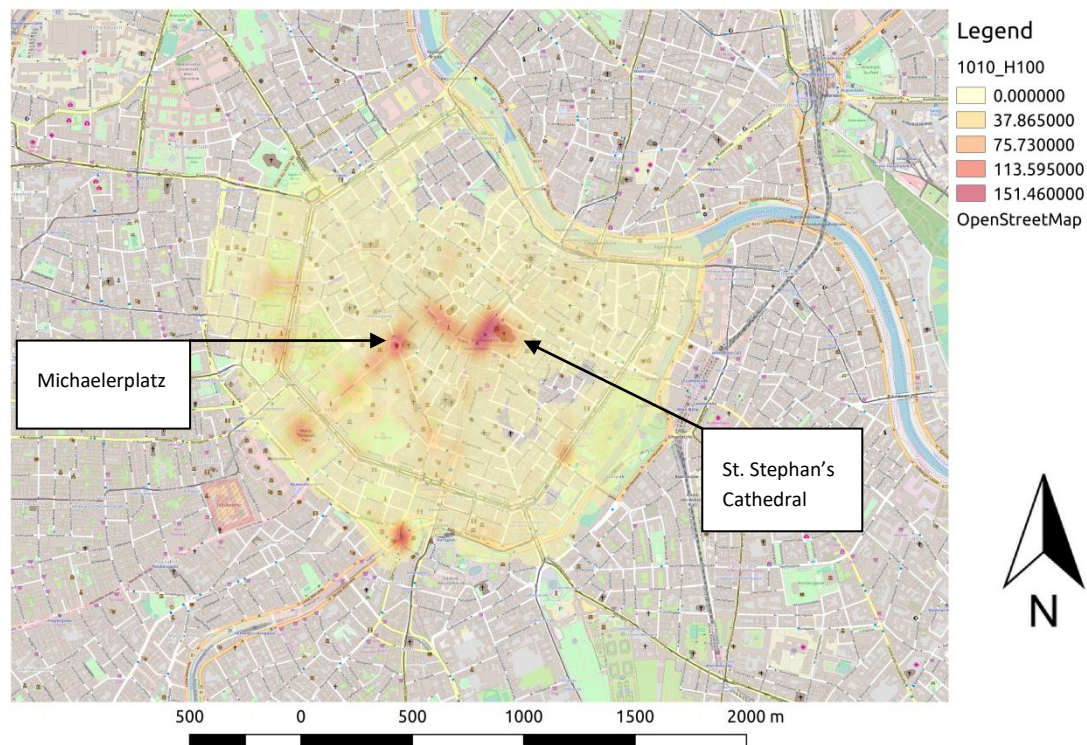


Illustration 19: Sole heat map 1st district

5.2.2 3rd district “Landstraße”

The second hot spot of the final model can be found in the 3rd district. In comparison to the hot spot in the first district it can be seen that this spot is already more local. Illustration 20 shows that this spot is located in the north west of the 3rd district.

Illustration 21 once again presents a more detailed look into the area. The point radius for this map was 200m. Since the area of the 3rd district is larger than the one of the 1st a larger radius was chosen to get a better coverage of the area. Since the vast majority of images cluster around a few specific spots a bigger radius ensures more touching areas in parts with particular few images.

With the sole district map two hot spots can be detected – first one around the Belvedere Palace in the west and the second at the Hundertwasser House in the north. Once again two well-known tourist spots.

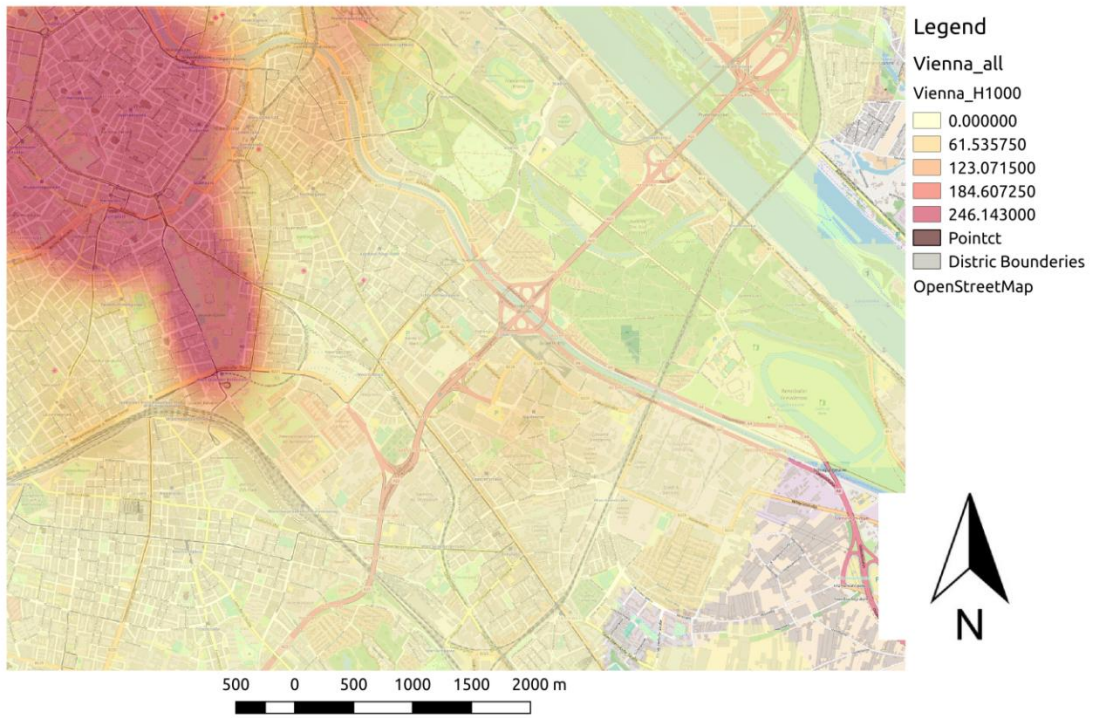


Illustration 20: Enlarged hotspot 3rd district

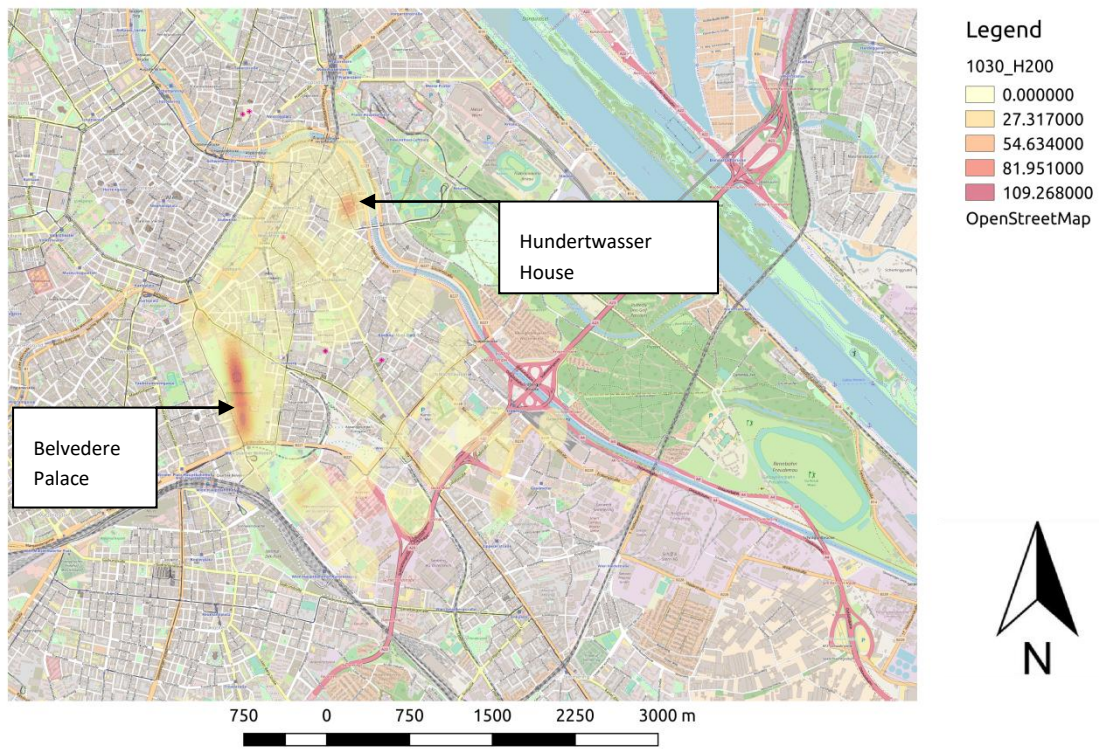


Illustration 21: Sole heat map 3rd district

5.2.3 13th district “Hietzing”

The third hot spot of the final model is located in the 13th district. Although there are about 1200 points in the area the overall coverage is not very good. The 13th district is one of the largest in Vienna and, with the “Lainzer Tiergarten” within its border, has a big area of recreation. Areas offside of agglomerations tend to produce fewer images.

Most of the points are assembled around the Schönbrunn Palace. Illustration 22 shows the hot spot of the overall map and Illustration 23 the heat map for the single district. For the single heat map a radius of 300m was chosen. Since the area of this district is quite large compared to the previous ones, a larger radius was required to find potentially new hot spots.

Although the influence radius was reduced to 300m only one hot spot was created and shows the same location as the overall heat map on a more detailed scale.

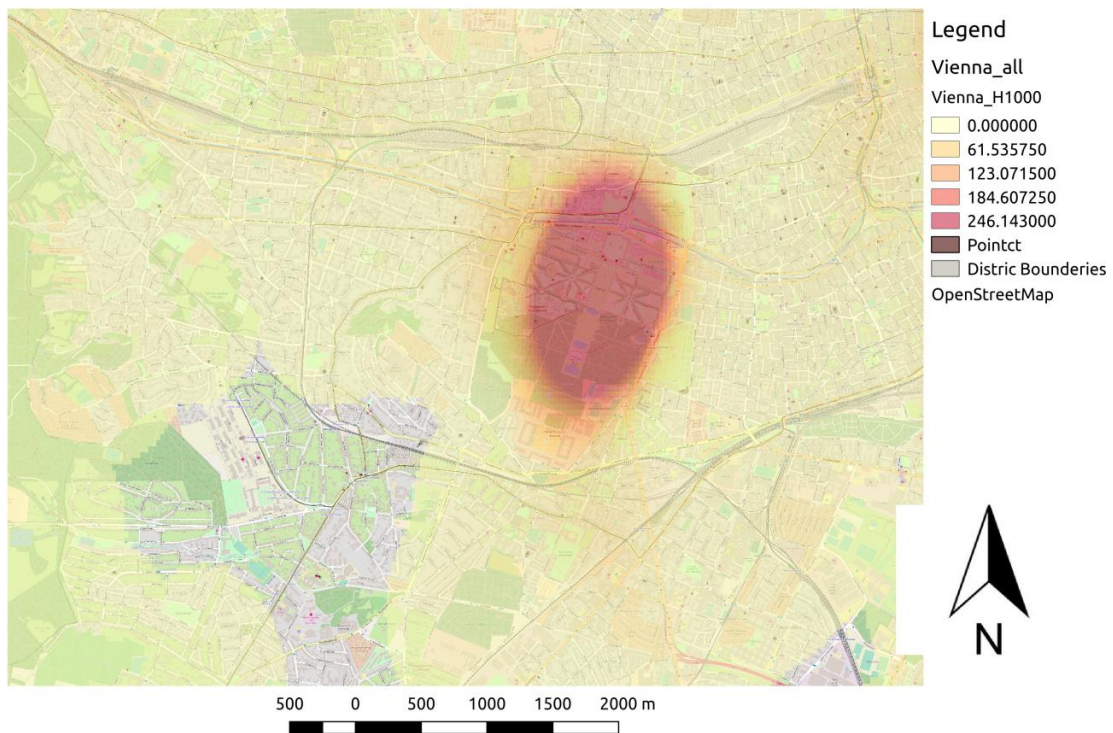


Illustration 22: Enlarged hotspot 13th district

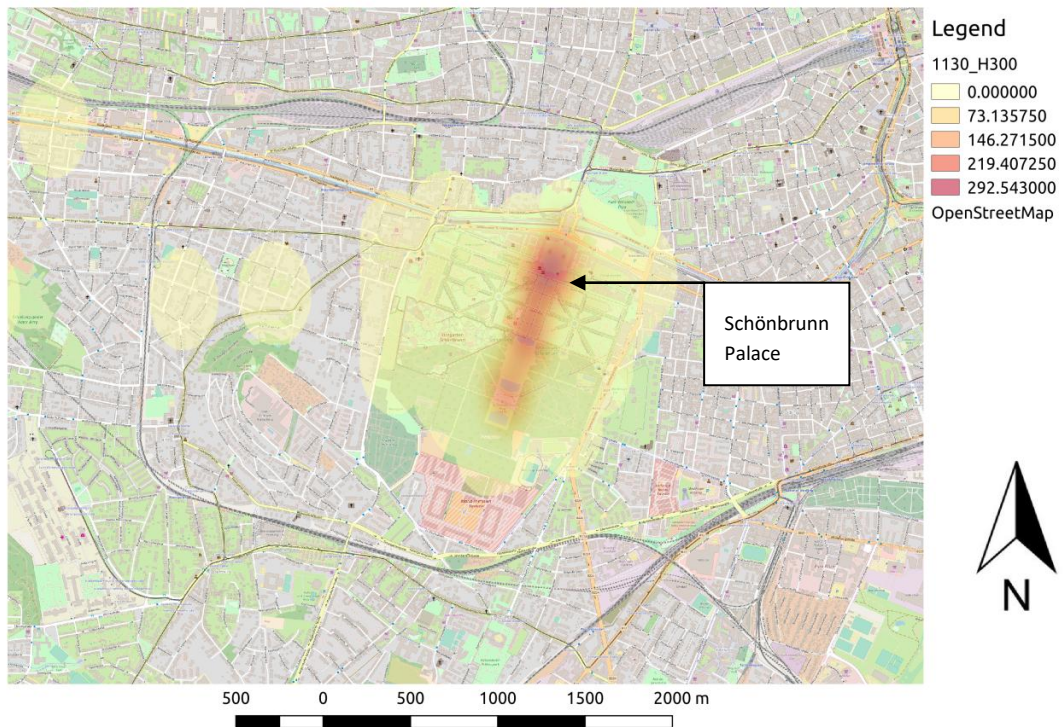


Illustration 23: Sole heat map 13th district

All in all it can be found, that a single heat map of the hot spots gives a detailed look of where the images are taken. Every hot spot found in the sole heat map is at or nearby a tourist hot spot.

6 Validation

In the final chapter the created VGI image density model will be compared with an actual real estate price model. By doing this it should be determined if the generated indicator resulting out of the density model is able to stand up to the claim that it correlates with real estate prices. The value of the indicator is the photo density and will be further on denoted as photo variable.

This is done by DataScience Service GmbH. In Chapter 1.3 one can find a brief introduction of the development of a real estate model by this company. The company assesses house prices by multilevel structured additive regression (Brunauer et al. 2013) and spatial heterogeneity (Helbich et al. 2013).

These two studies describe how their models and algorithm were created and which indicators are used. The result of this work was a model for Vienna and its catchment area and an overall model for Austria's real estate prices.

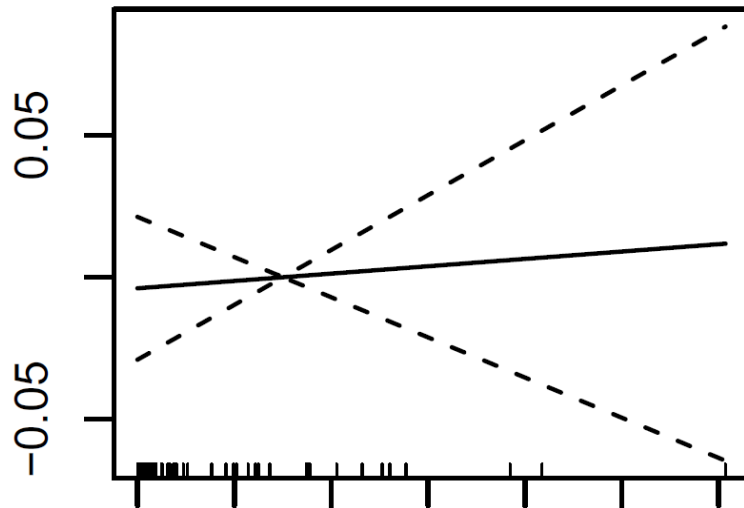
6.1 Verification set up

The real estate model by DataScience is dependent on several factors which flow into it. Each of the factors was checked against the price trend of the actual cost of properties. By this it was figured out which of the variables correlate with the price trend and which don't. The idea is to find indicators that correspond with the price trend in order to use these and their temporal changes to get a more precise model and to create future predictions.

There are some indicators which will always factor in the real estate price. Especially those indicators which are describing the object itself, like the age of the object, if it has a balcony or a garden and also interior factors e.g. the conditions of the floor or the heating system. But there are also more external factors which can determine the value of a property. There are the more obvious ones like the infrastructural situation around the object, e.g. the road connection, public transport and shopping areas and there are less obvious ones like power supply.

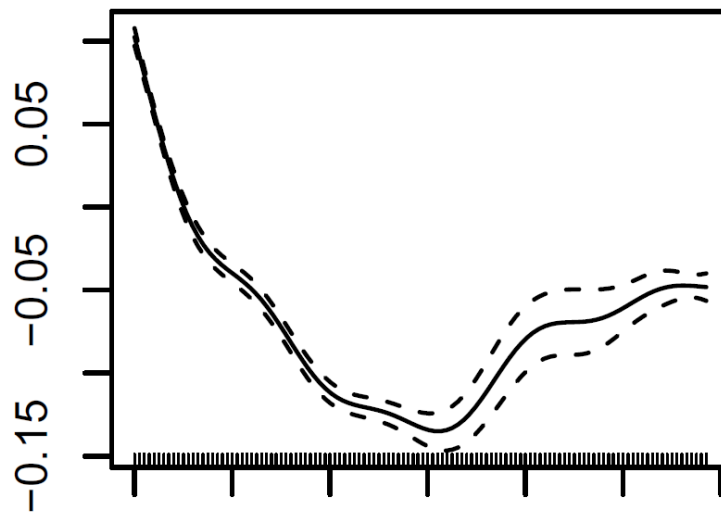
When checking indicators against the price model it is not always sensible to use all available indicators at the same time. Before the valuation it must be determined which indicators could influence each other and which have no effect on the indicator one wants to check. So when the photo variable will be checked indicators like the availability of drug stores will have no effect on the variable.

To get an idea how indicators perform Illustration 24 shows a bad performing indicator and Illustration 25 good performing one.



In_index_2

Illustration 24: Bad performing indicator



age_input

Illustration 25: Good performing indicator

The dashed lines in Illustration 24 and Illustration 25 delimit the range of the calculated value, represented by the solid line. Illustration 24 shows that the indicator is not corresponding with the price model. The cluster between the model and the actual curve is too wide spread. Furthermore if it is possible to put a straight line between the dashed lines,

then the effect of the indicator is not significant. In contrast to this Illustration 25 displays a very well performing indicator. This is the age indicator which was mentioned before. It can be seen that the indicator performs very well against the model. The spread between the curve and the model is very small and the cluster is moving along with the actual curve.

For a more detailed look into development of the real estate price model by DataScience it is recommended to take a look at the presented papers Brunauer et al. [2013] and Helbich et al. [2013]. Further readings regarding the methodical refinement of the model are found in Brunauer et al. (2017) is suggested. An insight on how the price model works is provided by Brunauer (2016).

6.2 Reality claim of the photo variable

The real estate price model used for the validation uses almost exclusively indicators which are describing the object. Next to them only few extern indicators like surrounding noise at a property were added. The reason for this was to check if extern indicators next to the vital object indicators have an effect on the photo variable.

When the photo variable was feed into the real estate model a strong correlation to the real estate price was detected. The resulting curve of the photo variable behaves well compared to the real estate model. As seen in Illustration 26 the cluster of the model next to the curve of the photo variable is quite small and moves along with the curve. Furthermore the behavior of the photo variable within the model was strictly monotonically increasing almost linear increasing. Additionally it can be observed that the uncertainty of the curve stays quite small even in parts of the curve with a low observation count. Also it turns out the extern indicators hardly change the impact of the photo variable.

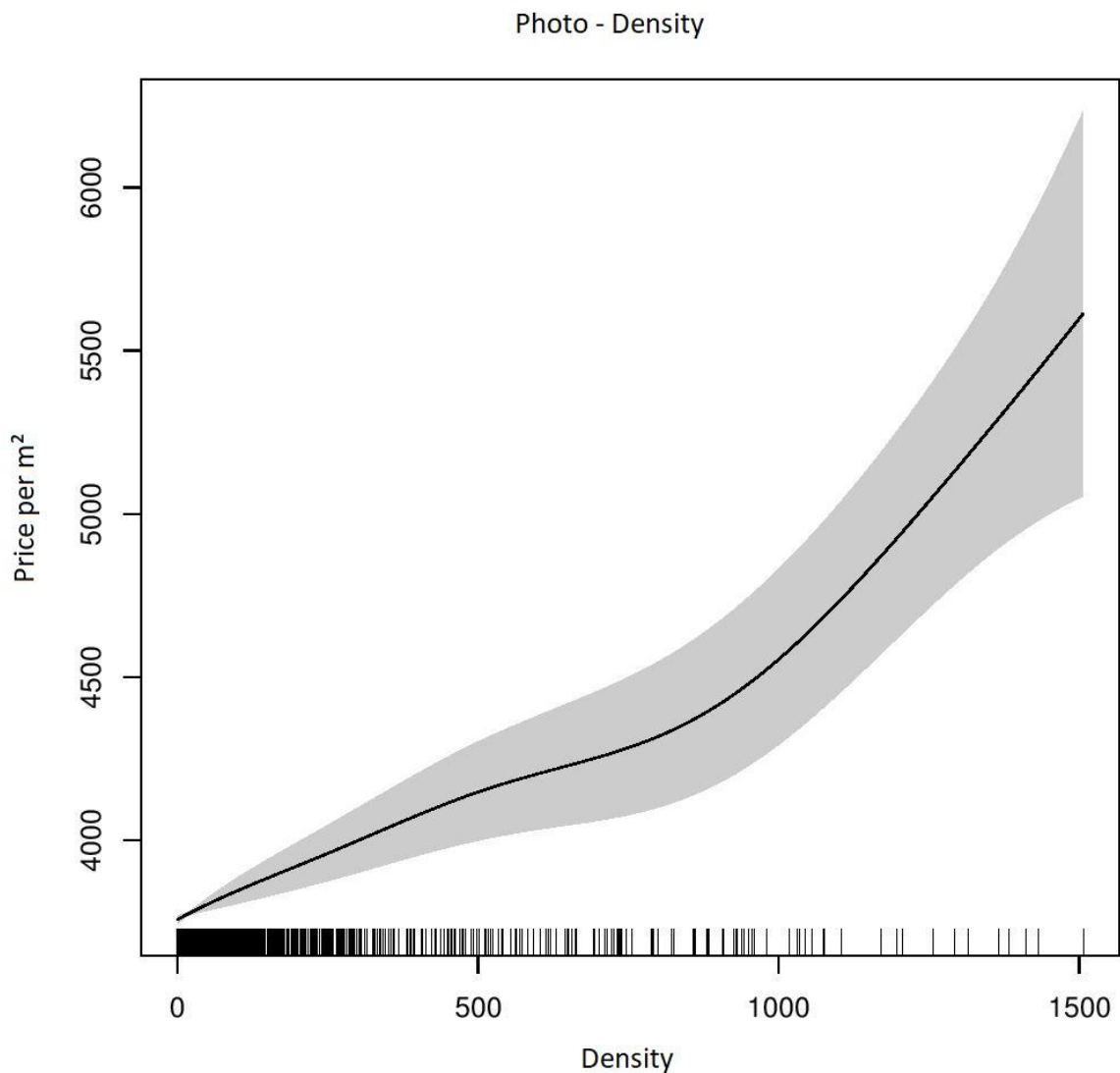


Illustration 26: Photo variable trend within the model

This correlation between the photo variable and the real estate price model imply that the amount of images taken on a specific spot are actually linked to property prices. A higher density of VGI images shows higher real estate prices and vice versa.

There are two possible reasons for this correlation:

- First of all, the original claim that an area which appears to be more beautiful produces more images is correct and therefore the photo variable is an indicator of beauty.
- The second reason could be that property prices are high in close proximity to tourist hotspots.

7 Summary and further research

In the course of this study a different approach to the work of VGI data was presented. First of all a possible way to categorize social media images was developed. This was done to create a classification which is able to be used to train a neural network. Using deep learning for classification will drastically decrease the time for classification. Furthermore a look into data quality of location metadata was done. It was demonstrated why GPS metadata is more trustworthy than geotag data. Based on this knowledge a solution was presented how to extract GPS data from an Exif-file of an image. In order to reduce storage space a method was presented to extract Exif information without the need of the original image. To secure all vital information it was shown how to add the extracted information of an original image to a smaller version of the same file. To ensure that these methods could be used for further research two scripts were created in such a way that they could be used for each area around the world with just some minor modifications. The finally presented method, creation of a heat map, was a standard procedure and was documented for recreation process.

The aim of this study was not only to provide automated methods for classification and VGI data acquisition but also to investigate, if there was a link between social media output and real estate prices. The final chapter of this research shows that the assumption in terms of the connection of high image density and high real estate prices can be considered as correct since there is a strong correlation between photo variables and high real estate prices. Furthermore, earlier research showed that areas with more social media output appear to be more beautiful. Therefore, it is highly probable that the photo variable serves as an indicator of beauty.

Before the validation of the heat map there was the presumption that the photo indicator could only detect tourist hot spots. But after the comparison to the real estate model it turns out that there is a correlation to real estate prices. Due to this strong correlation the result were highly satisfactory and open up the path for using this approach for further research.

Follow up research:

Since the time an image was taken is saved as well, it would be possible to create heat maps for certain time spans. Then, if the same class width and color scheme are used, it would be possible to compare these maps and see if there is a change of the hot spots within a year or even on bigger scale like five years. There would be solid data for at least the last 5 years, maybe even for the last decade.

A map set up like this could help to improve quality of life. With the information gained from these maps could lead to the information where to find highly frequented locations and how they change on course of a year. This could help to improve seasonal planning of tourist

attractions, but could lead to better time management for construction works, e.g. road construction or the renovation of tourist attractions as well. Furthermore, this attempt could be used to build dynamic maps of larger cities and therefore to have a better visibility of the hot spot movement. Or simply help the city to plan recreation projects in order to level up the overall beauty of the city

In order to use the CNN for further real estate price models, the created CNN should be able to handle all cities within middle and northern Europe, probably also US cities as well. Concerning southern European or South American cities the CNN has to be extended. Since the color scheme of images is important it could be that these cities are much brighter in the color composition than cities which lie further north. Therefore it could happen that valid images are falsely classified as non-valid. This assumption is based on the knowledge gained during the buildup of the CNN used in this work.

The same applies to country sides. The CNN has to be manipulated as well. We have to keep in mind as well that a different set of events must be set for filtering especially in Austria with its winter tourism. Winter tourism creates events, like ski slope or ski hut pictures that also need to be filtered in order to obtain the integrity of the CNN.

With extensions like these a CNN should be able to handle images from all over Austria. This would create the possibility to generate a density model for the whole country. But keep in mind that aside from larger cities or popular tourist destinations the photo density could be pretty low. In Austria are large amounts of pretty small villages or untouched nature which are most likely will not produce images at all.

The methods and techniques presented in this study should serve as a solid base for further research concerning VGI images, their classification and location data extraction. In addition they present a different way to create new indicators for real estate price development.

8 References

- Goodchild, M. F. (2007). *"Citizens as sensors: The world of volunteered geography."* GeoJournal. 69 (4), 211-221.
- Alivand, M. und Hochmair, H. (2013). *"Extracting scenic routes from VGI data sources."* In: GEOCROWD '13 Proceedings of the Second ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information, ACM, S. 23-30
- Brunauer, W., Lang, S. and Umlauf, N. (2013): *"Modeling House Prices using Multilevel Structured Additive Regression."* Statistical Modelling13(2), pp. 95-123.
- Helbich, M., Brunauer, W., Vaz, E., Nijkamp, P. (2013): *"Spatial Heterogeneity in Hedonic House Price Models: The Case of Austria."* Urban Studies 51(2), pp. 1¿22.
- Hochmair, H. (2010) *"Spatial Association of Geotagged Photos with Scenic Locations."* In: Car/Griesebner(Eds.) Proceedings of GI_Forum, Wichmann, S. 91-100.
- Instagram on Wikipedia* (2017). Available at: <https://en.wikipedia.org/wiki/Instagram> (Accessed: 05.12.2017)
- Flickr on Wikipedia* (2017). Available at: <https://en.wikipedia.org/wiki/Flickr> (Accessed: 05.12.2017)
- API on Wikipedia (2017). Available at: Flickr on Wikipedia (Accessed: 05.12.2017).
- Flickr API on Flickr.com* (2017). Available at: <https://www.flickr.com/services/api/> (Accessed: 05.12.2017)
- Schmidhuber, J. (2015). *"Deep Learning in Neural Networks: An Overview"*. Neural Networks. 61: 85–117.
- Hijazi, S., Kumar, R., Rowen, C., (2015). *"Using Neural Networks for Image Recognition"*. Tech. Rep., 2015. [Online]. Available: https://ip.cadence.com/uploads/901/TIP_WP_cnn_FINAL-pdf
- Guerra L, McGarry L.M., Robles V., Bielza C., Larrañaga P., Yuste R. (2010) *"Comparison between supervised and unsupervised classifications of neuronal cell types: a case study."* Dev. Neurobiol. 71(1), 71-82 (2011)
- Mohri, M., Rostamizadeh, A., Talwalkar, A. (2012) *"Foundations of Machine Learning"* eBook The MIT Press p.7
- Deep Learning on Wikipedia* (2017). Available at: https://en.wikipedia.org/wiki/Deep_learning (Accessed: 05.12.2017)

Convolutional Neural Networks on Wikipedia (2017). Available at:
https://en.wikipedia.org/wiki/Convolutional_neural_network (Accessed: 05.12.2017)

Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. (2015). "Going deeper with convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

Overfitting on OxfordDictionaries.com Statistics (2017). Available at:
<https://en.oxforddictionaries.com/definition/overfitting> (Accessed: 05.12.2017)

QGIS on Wikipedia (2017). Available at:<https://en.wikipedia.org/wiki/QGIS> Accessed:
(05.12.2017)

Hauff C. (2013). "A study on the accuracy of Flickr's geotag data." SIGIR '13, pp. 1037-1040.

TensorFlow (2017). Available at: <https://www.tensorflow.org/> (Accessed: 05.12.2017)

TensorFlow for poets (2017). Available at:
<https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/index.html?index=..%2F..%2Findex#0> (Accessed: 05.12.2017)

GeoTIFF at Wikipedia (2017). Available at: <https://en.wikipedia.org/wiki/GeoTIFF> (Accessed:
05.12.2017)

Brunauer, W., Feilmayr, W., Wagner, K. (2017). "Der österreichische Wohnimmobilienpreisindex: Methodische Verfeinerung 2017".

Transl.: "The Austrian real estate price index: methodical refinement 2017". At the time this thesis was written the paper was not yet released and will be published in German.

Brunauer, W. (2016). "Die modellhafte Ermittlung von Vergleichswerten". ÖVI News 03/2016, pp. 8-10. Transl.: The model-like development of comparative values.

The article is written in German.

9 Illustration directory

Illustration 1: 4KStogram search by location	5
Illustration 2: 4KStogram data view	5
Illustration 3: Kunsthistorisches Museum by Sandor Somkumi	8
Available at: https://www.flickr.com/photos/somkuti/34370894934/ (License: Creative Commons) (Accessed: 05.12.2017)	
Illustration 4: Scheme of a CNN	11
Author unknown; seen at: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/ (Accessed: 05.12.2017)	
Illustration 5: Chart view of curve fitting	12
Author unknown; seen at: http://blog.algotrading101.com/wp-content/uploads/2016/01/Chart-view-of-curve-fitting.png from http://blog.algotrading101.com/design-theories/what-is-curve-fitting-overfitting-in-trading/ (Accessed: 05.12.2017)	
Illustration 6: Adding own classes	13
Illustration 7: Geotag Assembly Point Facebook	25
Illustration 8: Creating layer form text file; QGis	40
Illustration 9: Coordinate System Selector; QGis	41
Illustration 10: Raw data and boundaries of Vienna; QGis	41
Illustration 11: Select by location tool; QGis	42
Illustration 12: Data points within Vienna; QGis	42
Illustration 13: Heat map Plug-in; QGis	43
Illustration 14: Layer preparation; QGis	43
Illustration 15: Finished Heat map; QGis	44
Illustration 16: Image falsely classified as valid	45
Illustration 17: Final Model for Vienna	50
Illustration 18: Enlarged hotspot 1 st district	51
Illustration 19: Sole heat map 1 st district	52
Illustration 20: Enlarged hotspot 3 rd district	53
Illustration 21: Sole heat map 3 rd district	53
Illustration 22: Enlarged hotspot 13 th district	54

Illustration 23: Sole heat map 13th district..... 55

Illustration 24: Bad performing indicator..... 57

Illustration 25: Good performing indicator 57

Illustration 26: Photo variable trend within the model 59

All images used for the examples are public images from Instagram. All rights of the images belong to Instagram and therefore to the Facebook Inc.

Example 1 18

Creator: annush.ka7 Available at: <https://www.instagram.com/p/BYgZwGHIB8q/> (Accessed: 05.12.2017)

Example 2 18

Creator: tristanhbe; Available at: <https://www.instagram.com/p/BYIAtZOjHGo/> (Accessed: 05.12.2017)

Example 3 18

Creator: angela_travels; Available at: <https://www.instagram.com/p/BYlxCUVgLh8/> (Accessed: 05.12.2017)

Example 4 18

Creator: 2____jin ; Available at: <https://www.instagram.com/p/BYI9DoKlo6m/> (Accessed: 05.12.2017)

Example 5 18

Creator: vzlabeer; Available at: <https://www.instagram.com/p/BYI8DFhn2m8/>**Fehler! Textmarke nicht definiert.** (Accessed: 05.12.2017)

Example 6 18

Creator: agnesonthecloud; Available at: <https://www.instagram.com/p/BYnLP9hjTXL/> (Accessed: 05.12.2017)

Example 7 19

Creator:lubosaugustin; Available at: <https://www.instagram.com/p/BYIzycIH9j/> (Accessed: 05.12.2017)

Example 8 19

Creator: swifstyle; Available at: <https://www.instagram.com/p/BYi1HjIhTGZ/> (Accessed: 05.12.2017)

Example 9	19
Creator: andreagerak; Available at: https://www.instagram.com/p/BYgtidwFOkV/ (Accessed: 05.12.2017)	
Example 10	20
Creator: aspirinia; Available at: https://www.instagram.com/p/BYmAiDQD3mY/ (Accessed: 05.12.2017)	
Example 11	20
Creator: maijano Available at: https://www.instagram.com/p/BYgys60AdEM/ (Accessed: 05.12.2017)	
Example 12	20
Creator: jeremy.gibert; Available at: https://www.instagram.com/p/BYge3PfhXy_/ (Accessed: 05.12.2017)	
Example 13	20
Creator: hanif.sh21 ; Available at: https://www.instagram.com/p/BYgwzangFrU/ (Accessed: 05.12.2017)	
Example 14	20
Creator: gentlemensflair; Available at: https://www.instagram.com/p/BYgmYwTAWht/ (Accessed: 05.12.2017)	
Example 15	20
Creator: discouture; Available at: https://www.instagram.com/p/BYgYaWbAr_o/ (Accessed: 05.12.2017)	
Example 16	21
Creator: shreyaclaire; Available at: https://www.instagram.com/p/BYgMpBan4Pa/ (Accessed: 05.12.2017)	
Example 17	21
Creator: renyswelt; Available at: https://www.instagram.com/p/BYfOI0Fhbd7/ (Accessed: 05.12.2017)	
Example 18	21
Creator: weiss.olivia; Available at: https://www.instagram.com/p/BYgHy9AFLlc/ (Accessed: 05.12.2017)	
Example 19	22

Creator: martin_on_point; Available at: <https://www.instagram.com/p/BYfeXDxlX1P/>
(Accessed: 05.12.2017)

Example 20 22

Creator: ___shygirl___ ; Available at: <https://www.instagram.com/p/BYfxJc4H2Ue/>
(Accessed: 05.12.2017)

Example 21 22

Creator: sara.svobodova; Available at: https://www.instagram.com/p/BYfdvspg5p_/
(Accessed: 05.12.2017)

Example 22 22

Creator: luvtheshop; Available at: <https://www.instagram.com/p/BYnK751HRTr/> (Accessed:
05.12.2017)

Example 23 22

Creator: aylin.sha; Available at: <https://www.instagram.com/p/BYnE7c4FkOi/>**Fehler!**
Textmarke nicht definiert.
(Accessed: 05.12.2017)

Example 24 22

Creator: elvisalisonvienna; Available at: <https://www.instagram.com/p/BYm2-ZYgg7H/>
(Accessed: 05.12.2017)

Example 25 23

Creator: slimusgm; Available at: <https://www.instagram.com/p/BYmzRJtDX-a/> (Accessed:
05.12.2017)

Example 26 23

Creator: bythewayando ; Available at: <https://www.instagram.com/p/BYmwi9mAklX/>
(Accessed: 05.12.2017)

Example 27 23

Creator: mariaflor_11; Available at: <https://www.instagram.com/p/BYmRI-LHxTI/> (Accessed:
05.12.2017)

Example 28: Valid Food 24

Creator: explorophile; Available at: <https://www.instagram.com/p/BYmHi2cHp83/>
(Accessed: 05.12.2017)

Example 29: Valid Meme..... 24

Creator: safia444 ; Available at: <https://www.instagram.com/p/BYkX26NldJt/> (Accessed: 05.12.2017)

Example 30: Valid Photo-shoot 24

Creator: solomiakolodnytska; Available at: <https://www.instagram.com/p/BYkV8wfAufM/> (Accessed: 05.12.2017)

Example 31: Valid Selfie 24

Creator: macajarao; Available at: <https://www.instagram.com/p/BYjqtM5FIyN/> (Accessed: 05.12.2017)

Example 32: Valid stuff..... 24

Creator: protogane_gte ; Available at: <https://www.instagram.com/p/BYec23aFsns/> (Accessed: 05.12.2017)

Table 1: Image Derivation Instagram 29

Table 2: First test set up 31

Table 3: Configurations of the training set ups and their accuracy 31

Table 4: Image distribution of the final set up 32

Table 5: Summery of viewed image data 47

Table 6: Distribution of images per district..... 48

Table 7: Statistical characteristics of the points 48

Table 8: Derivation for 5 classes..... 69

Table 9: Derivation for 4 classes..... 69

Table 10: Derivation for 3 classes 69

Table 11: Derivation for 2 classes 70

10 Attachment

10.1 Image derivations of the CNNs

Table 8: Derivation for 5 classes

Class	Images	Test set	Total	Percentage
Indoor	1620	103	1723	17,01
Photo-shoot	1770	157	1927	19,03
Random	1150	97	1247	12,31
Stuff	1700	147	1847	18,24
Valid	3100	284	3384	33,41
				100,00
Total	9340	788	10128	
Original Downloaded			11000	
Useless Data			7,93	

Table 9: Derivation for 4 classes

Class	Images	Test set	Total	Percentage
Indoor	1620	103	1723	17,01
Photo-shoot	1770	157	1927	19,03
Random	2850	244	3094	30,55
Valid	3100	284	3384	33,41
				100,00
Total	9340	788	10128	
Original Downloaded			11000	
Useless Data			7,93	

Table 10: Derivation for 3 classes

Class	Images	Test set	Total	Percentage
Photo-shoot	3390	260	3650	36,04
Random	2850	244	3094	30,55
Valid	3100	284	3384	33,41
				100,00
Total	9340	788	10128	
Original Downloaded			11000	
Useless Data			7,93	

Table 11: Derivation for 2 classes

Class	Images	Test set	Total	Percentage
Non-valid	6240	504	6744	66,59
Valid	3100	284	3384	33,41
				100,00
Total	9340	788	10128	
Original Downloaded			11000	
Useless Data			7,93	

10.2 Scripts

10.2.1 Downloading script

```
import flickrapi
import numpy as np
import urllib
import os
import piexif
from PIL import Image
import exifread

flickrAPIKey = "380681cf13e54df65df45465df42e4bf584" # API key
flickrSecret = "eac554d5f4d32f07 " # shared "secret"

fapi = flickrapi.FlickrAPI(flickrAPIKey, flickrSecret, format='parsed-json')

img_dir = "/home/k-man/tf_files/gps/"

timeskip = 604800 #one_week
mintime = 1467244800 # 30.06.2016
maxtime = mintime+timeskip
endtime = 1498780800 #30.06.2017

extras='geo,url_sq,url_t,url_s,url_q,url_m,url_n,url_z,url_c,url_l,url_o'

time_steps = np.arange(mintime, endtime, timeskip)
time_steps = np.append(time_steps, endtime)

a = 1

for i in range(0, len(time_steps)-2):

    mintime = time_steps[i]
    endtime = time_steps[i+1]

    print "=====
    print "STARTIME %s" % str(mintime)
```

```

print "ENDTIME %s" % str(endtime)

    #INITIAL QUERRY TO GET AVAIL PAGES
    rsp = fapi.photos.search(api_key=flickrAPIKey,
    ispublic="1",
    media="photos",
    per_page="250",
    page=str(curr_page),
    bbox="16, 47, 17, 49",
    min_upload_date=str(mintime),
    max_upload_date=str(endtime),
    extras=extras)

avail_pages = rsp['photos']['pages']

print "AVAILABLE PAGES: %i" % (avail_pages)
print "-----"
    #LOOP THROUGH EACH PAGE
    page_nrs = np.arange(0, avail_pages+1)

    forcurr_page in page_nrs:

        print "CURRENT PAGE: %i " % (curr_page)

        rsp = fapi.photos.search(api_key=flickrAPIKey,
        ispublic="1",
        media="photos",
        per_page="250",
        page=str(curr_page),
        bbox="16, 47, 17, 49",
        min_upload_date=str(mintime),
        max_upload_date=str(endtime),
        extras=extras)

    all_photos = rsp['photos']['photo']
    print "NR OF PHOTOS: %i" % (len(all_photos))

```

```

        i = 1

for photo in all_photos:

photo_id = str(photo['id'])
down = False

print i, photo_id,
        i += 1

        # Check if Exif is available
try:
        photo_exif = fapi.photos.getExif(api_kay=flickrAPIKey, photo_id=photo_id)

except:
pass

exif = photo_exif['photo']['exif']

        #Check for GPS location data
ifexif:

avail_tags = []

for tags in exif:
avail_tags.append(tags['tag'])

if "GPSLatitude" in avail_tags:

for tags in exif:

tag = tags['tag']

if tag == "GPSLongitude":

lon_str= tags['raw']['_content']
elem = lon_str.split(" ")

```



```

lon_deg = int(elem[0])
lon_min = int(elem[2].replace("'", ""))
lon_sec = int(float(elem[3].replace("'", ""))*100)

#lon = lon_deg + lon_min / 60. + lon_sec /3600.

if tag == "GPSLatitude":

print " GPS: YES"

lat_str= tags['raw']['_content']
elem = lat_str.split(" ")

lat_deg = int(elem[0])
lat_min = int(elem[2].replace("'", ""))
lat_sec = int(float(elem[3].replace("'", ""))*100)

#lat = lat_deg + lat_min / 60. + lat_sec / 3600.

photo_info = fapi.photos.getInfo(api_key=flickrAPIKey, photo_id=photo_id)

# Get Photo information
img_time= photo_info['photo']['dates']['taken']

photo_sizes = fapi.photos.getSizes(api_key=flickrAPIKey, photo_id=photo_id)
avail_sizes = photo_sizes['sizes']['size']

size_labels = []

for sizes in avail_sizes:
size_labels.append(sizes['label'])

if "Original" in size_labels:

#if original use the third last image size
#to reduce disk space and download time

```

```

size = avail_sizes[-2]

source = size['source']

img_name = source.split("/")[-1]
img_path = os.path.join(img_dir, img_name)

try:
urllib.urlretrieve(source, img_path)

down = True

except:
down = False

else:
    #if original not available use the second last image size
size = avail_sizes[-1]

source = size['source']

img_name = source.split("/")[-1]
img_path = os.path.join(img_dir, img_name)

urllib.urlretrieve(source, img_path)

try:
urllib.urlretrieve(source, img_path)
down = True

except:
down = False

if down:

im = Image.open(img_path)
#Add new Exif

```

```

gps_ifd = {
                piexif.GPSIFD.GPSLatitude: ((lat_deg,1), (lat_min,1), (lat_sec,100)),
                piexif.GPSIFD.GPSLongitude: ((lon_deg,1), (lon_min,1), (lon_sec,100)),
                piexif.GPSIFD.GPSDateStamp: img_time
            };

exif_dict = {"GPS":gps_ifd}
exif_bytes = piexif.dump(exif_dict)

im.save(img_path, exif=exif_bytes)

                f = open(img_path)
tags = exifread.process_file(f)
f.close()

else:

print " GPS: NO"
else:
print "NO EXIF"

```

10.2.2 Test script

```

import tensorflow as tf, sys
import os

work_class = sys.argv[1]

work_dir = os.path.join("/tf_files/", work_class)
# docker_dir = "/tf_files/Classes/"
result_csv = os.path.join(work_dir, "fail.csv")

with open(result_csv, "wb") as out_csv:
    out_csv.write("PATH\n")

```

```

# Unpersists graph from file
with tf.gfile.FastGFile("/tf_files/retrained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    _ = tf.import_graph_def(graph_def, name='')

for files in os.listdir(work_dir):
    if files.endswith(".jpg"):

        image_path = os.path.join(work_dir, files)
        print "===== <"
        print image_path

            #with tf.Graph().as_default():
            # Read in the image_data
        image_data = tf.gfile.FastGFile(image_path, 'rb').read()

            # Loads label file, strips off carriage return
        label_lines = [line.rstrip() for line
            in tf.gfile.GFile("/tf_files/retrained_labels.txt")]

with tf.Session() as sess:
    # Feed the image_data as input to the graph and get first prediction

        softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

    predictions = sess.run(softmax_tensor, \
                            {'DecodeJpeg/contents:0': image_data})

    # Sort to show labels of first prediction in order of confidence

        top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]

    for node_id in top_k:
        human_string = label_lines[node_id]

```

```

ifhuman_string == "non valid":

score = predictions[0][node_id]

if score > 0.9:

print('ok')

else:
print('Wrong class')
print(image_path

                                f = open(image_path, 'rb')
tags = exifread.process_file(f)

out_csv.write("%s\n" % (image_path))

```

10.2.3 Classification script

```

import tensorflow as tf, sys
import os
import exifread

work_class = sys.argv[1]

work_dir = os.path.join("/tf_files/", work_class)
#docker_dir = "/tf_files/Classes/"
result_csv = os.path.join(work_dir, "classified.csv")

with open(result_csv, "wb") as out_csv:

    out_csv.write("LAT;LON;PATH;TIME\n")

    # Unpersists graph from file
    with tf.gfile.FastGFile("/tf_files/retrained_graph.pb", 'rb') as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f.read())
        _ = tf.import_graph_def(graph_def, name='')

```

```

for files in os.listdir(work_dir):
    if files.endswith(".jpg"):

        image_path = os.path.join(work_dir, files)
        print "===== < <"
        print image_path

        # Read in the image_data
        image_data = tf.gfile.FastGFile(image_path, 'rb').read()

        # Loads label file, strips off carriage return
        label_lines = [line.rstrip() for line
                       in tf.gfile.GFile("/tf_files/retrained_labels.txt")]

        with tf.Session() as sess:
# Feed the image_data as input to the graph and get first prediction

            softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

            predictions = sess.run(softmax_tensor, \
                                   {'DecodeJpeg/contents:0': image_data})

# Sort to show labels of first prediction in order of confidence

            top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]

            for node_id in top_k:

                human_string = label_lines[node_id]

                if human_string == "valid":

                    score = predictions[0][node_id]

                    if score > 0.9:

                        print('%s (score = %.5f)' % (human_string, score))

```

```

f = open(image_path, 'rb')
tags = exifread.process_file(f)

GPSLatitude" in tags and "GPS GPSLongitude" in tags:
                                                    if "GPS

lat_list = tags["GPS GPSLatitude"].values

lat_deg = int(str(lat_list[0]))
lat_min = int(str(lat_list[1]))
lat_sec = str(lat_list[2]).split("/")
lat_sec = float(lat_sec[0]) / float(lat_sec[-1])
lat = lat_deg + lat_min / 60. + lat_sec / 3600.

lon_list = tags["GPS GPSLongitude"].values
lon_deg = int(str(lon_list[0]))
lon_min = int(str(lon_list[1]))
lon_sec = str(lon_list[2]).split("/")
lon_sec = float(lon_sec[0]) / float(lon_sec[-1])
lon = lon_deg + lon_min / 60. + lon_sec / 3600.

time = tags["GPS GPSDate"].values

out_csv.write("%.6f;%.6f;%s;%s;%.2f\n" % (lat, lon,
image_path, time, score))

else:
    print('%s (score = %.5f)' % (human_string, score))

```