CLASSIFYING SWISS CITIES BASED ON AERIAL IMAGERY

ELWIN FREUDIGER

Abstract. This project aims to explore the possibility of classifying cities based on Aerial imagery. The dataset used is $1km^2$ tiles taken from the *Swiss Topographical Office*. The model is trained using several color and texture features such as pixel distribution or contrast. Focusing on several cities in Switzerland, this project trains a Deep-neural-network model to achieve this. This study demonstrates the feasibility of such a model and can have a wide range of applications.

Key words. Classification, Deep Neural Networks, Aerial imagery, Topography, Image Features

1. Introduction. Aerial imagery is used in a wide variety of fields. The applications are almost endless, from military reconnaissance to weather forecasting. Extracting data and knowledge from this imagery is becoming more important and with it the need to develop neural processes to accomplish these tasks. This paper explores how machine learning techniques can be used to classify cities based on their satellite imagery. Using a dataset comprising of $1km^2$ tiles taken from the Swiss Topographical Office, the research will attempt to understand how a deep neural network may be trained to classify images using color and texture features. The addition of contextual environmental data will be attempted to determine how their addition may impact the overall accuracy. The objective of this paper is to leverage a machine learning model using image features to determine if classification is possible.

2. Research Question and Relevant Literature. This section aims to give the reader a better understanding of the source images, the features extracted and the API used.

2.1. Image source. All of the images used in this project were obtained thanks to the *Federal Office of Topography swisstopo*, their website allows anyone to download a mosaic of aerial images. These images are taken all across Switzerland and the edge of the country. Each image has a ground resolution of 10cm, except over the Alps where the ground resolution is 25cm. These tiles are $1km^2$, for this research, a lower download resolution of 2 meters was chosen. This choice was motivated by the need to save space. The files used were downloaded as a Cloud Optimized GeoTIFF, these types of files are widely used in Topography because, images can be stored with georeferencing metadata. The main image is stored as a $3 \times 500 \times 500$ array in an RGB format. The metadata contained in the file contains an Affine Transformation, with which it is possible to derive the tiles center coordinates.

It should be noted that the Coordinate Reference System (CRS) used here is the Swiss Coordinate System, known as $CH1903 + /LV95$, specially developed by Switzerland. It is similar to the lat./lon. system with the difference of being centered around an observatory around Bern and being expressed as East and North and incrementing in meters. The field of Topography is a vast and exciting field, unfortunately, it would be impossible to cover everything in this paper. More information regarding the Mosaic can be found in this PDF.

2.2. Feature extraction. The downloaded file will be a stacked RGB array. As a reminder, RGB is a way of displaying images where each pixel has a value for Red, Green, and Blue. A combination of these three primary colors can display any color. Values are always between 0 and 255. For example, a black pixel will have $RGB(0, 0, 0)$ values.

FIG. 1. *Sample satellite image*

Instead of using an array of pixels for classification, features will be extracted from the image and used as inputs for classification. This approach is preferred for several reasons, including efficiency, and improved results. Indeed, preliminary results using raw pixel arrays proved insufficient to accurately capture the differences between each city. Various literature suggests using color features and texture features for enhanced image classification performance. [1]. The features used will be the following:

- **Gray scale intensity histogram**: This histogram contains intensity values for the array as a grayscale image. It provides information about the distribution of pixel intensities.
- **RGB histogram**: This histogram helps determine which dominant colors appear in the image. For example, a predominantly green image filled with nature will show a higher intensity in green values, while an image of a lake will exhibit a high intensity in the blue values.
- **Mean pixel value**: This feature represents the average intensity of the pixels in the image.
- **Standard deviation of pixel values**: This feature measures the variation in pixel intensities, indicating how much contrast is present in the image.
- **Texture data**: Using a co-occurrence matrix, it is possible to extract features related to the texture of an image. This method is known as the Haralick features, named after Robert Haralick, who proposed these features for image classification, especially in the context of satellite imagery [2]. The features used here include Contrast, Correlation, Energy, and Homogeneity. These features capture the spatial relationship between pixels, providing valuable information about the texture patterns within the image.

By extracting these features, the classification model can better distinguish between different cities. This approach not only enhances the accuracy of the classification but also improves efficiency by reducing the size of the input data.

Using Figure 1 as a sample image, the following features can be extracted:

Mean pixel: 91.83, Standard deviation: 39.26, contrast: 1879.57, correlation: 0.32, energy: 0.01, homogeneity: 0.06. The distribution can be found in Figure 2

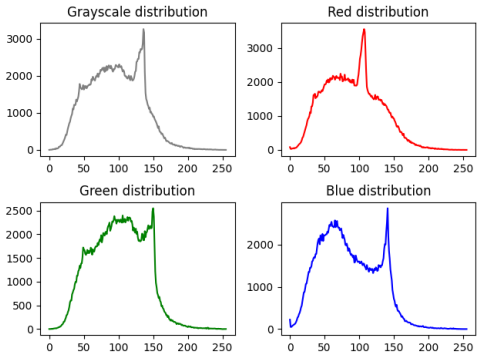

FIG. 2. *Color distribution*

2.3. Multiple Input Machine Learning Models. The Keras Functional API was used. This API is similar to the Sequential. But allows for the creation of more complex models that can handle for example multiple inputs, a feature used in this research. The data used is divided between single numbers and pixel distribution data, as such, a model capable of handling multiple inputs is necessary.

Creating a model using the Functional API is simple. The input layers are defined for each input type, each input is processed through its own set of layers. The layers from both sub-models are concatenated using the *layers.concatenate*() function. This combined layer integrates the features from both data types. Additional layers can be added to this concatenated layer before producing the output.

The Keras functional API handles the models as a Directed Graph of Layers. This feature also makes it possible to plot the model as a graph for better visualization.

A simple model in Functional Keras can be built with few lines of code:

```
import keras
from keras.models import Sequential
from keras.layers import Input, Flatten, Dense, Dropout

inputs = Input(shape= input_shape)
flat = Flatten()(inputs)
x = Dense(64, activation="relu")(flat)
x = Dropout(0.5)(x)
x = Dense(64)(x)
outputs = Dense(10)(x)

model = keras.Model(inputs=inputs, outputs=outputs)
```

3. Methodology. First, the data will be downloaded, cleaned and it's features extracted. Different variations will be attempted. First, the ten biggest cities by population will be downloaded.

This dataset is interesting from a political perspective, as it gathers the biggest cities by number of inhabitants. But it fails to take into account the size of cities, the number of observations by city, listed in Table 1 show that this dataset is not balanced enough.

Furthermore, Most cities selected are located in the Swiss Middleland as can be seen in Figure 3. This will make it harder to generalize the results obtained.

City	Number of Images
Lugano	131
Zurich	124
Lausanne	108
Winterthur	102
Bern	83
Luzern	69
St. Gallen	61
Basel	43
Biel	41
Geneva	32

TABLE 1
Number of images by city

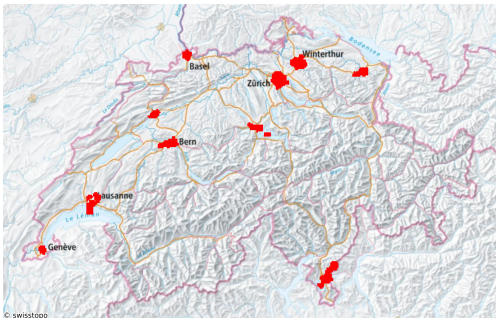

FIG. 3. *Images for the 10 biggest cities in Switzerland*

To avoid this, a different dataset that includes cities of similar size and in different regions of Switzerland was also created. The number of observation was more balanced as can be seen in Table 2

City	Number of images
Glarus	139
Lugano	131
Zurich	124
Nendaz	117
Sarnen	109
Lausanne	108
Winterthur	102
Bern	83
Schwyz	82
Chur	75

TABLE 2
Number of images for the balanced dataset

And the distribution in Figure 4 is better, capturing cities all across Switzerland.

Furthermore, to determine if the classification could be better when using additional environmental information, another dataset was created adding yearly averages

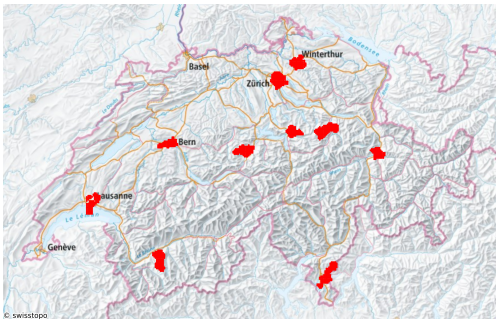

FIG. 4. *Mapping of the balanced dataset*

of *Temperature*, *Rainfall* and *Sunshine*. These maps seen in Figure 5 were also obtained from the *Federal Office of Topography*. A first classification will be performed without this data and it will then be added to see if added data can help the model.

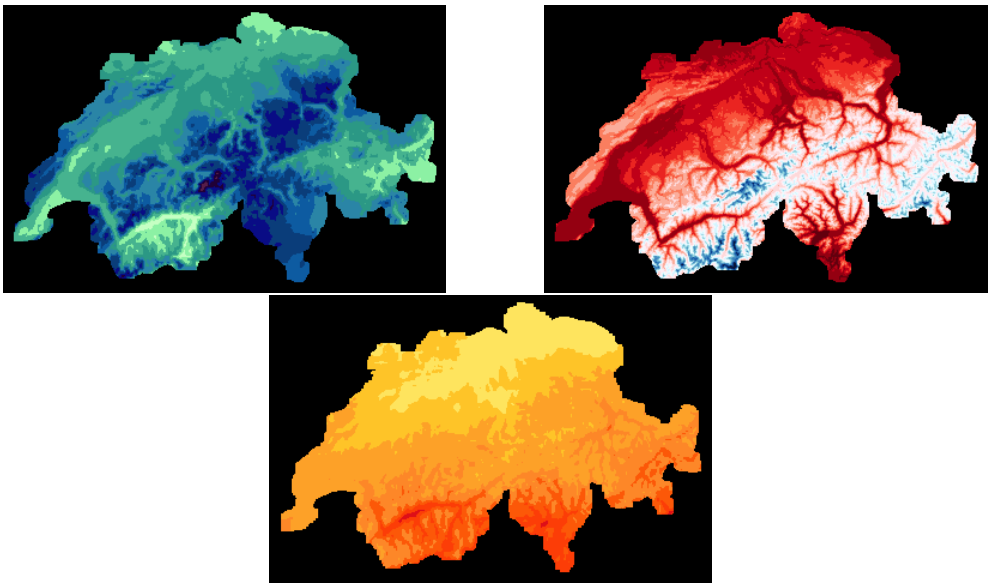

FIG. 5. *Rainfall, Temperature, and Sunshine levels in Switzerland*

4. Code implementation. The whole code can be found on Github: https://github.com/Elwin-Freudiger/Swisscities

The working tree is as follows:

```
/ (Swisscities)
├── src
│   ├── feature_extraction.py
│   └── Add_otherinfo.py
├── prediction
│   ├── balanced_city_predict.py
│   └── top10_city_predict.py
└── data
    └── city_link
```

```
    └── csv
        ├── Features_top10.csv
        ├── Features_balanced.csv
        ├── top10_with_info.csv
        └── balanced_with_info.csv
├── report
└── presentation.ipynb
```

4.1. Libraries used. Various libraries were used to accomplish this project:

- *requests*: to download content online based on a link
- *rasterio*: to open raster type data.
- *numpy*: for arrays.
- *OpenCV*: computer vision to transform images
- *skimage*: to compute the co-occurence matrix and extract texture features.
- *pandas and geopandas*: for dataframes and geo-dataframes.
- *alive_progress*: To create progress bars.
- *concurrent.futures*: To put processes in parallel
- *TensorFlow.Keras*: The Keras API to run neural networks.

4.2. Dataset description. As mentioned previously, the dataset was obtained through the Federal Office of Topography, on their website, it is possible to download tiles either one by one or regrouped by city or by Canton. Instead of downloading the images by hand, a CSV with download links is provided. The links for each city are in different files to allow more customization. The download of each file is performed using the request Library. The contents are then temporarily stored in the memory. This is done using the Rasterio library, each file is only stored while it's features are extracted which saves space. The center of the image is then inferred using the Affine Transformation, and expressed in the Swiss Coordinate System. Each feature is then extracted and added as a row to our dataset. As one may expect, this process takes time, and doing it for one thousand images would take too much time. Therefore, the process is threaded. This is done using the $concurrent.futures$ and $ThreadPoolExecutor()$.

The parallelization will repeat the extraction process and append them to a row in our dataset. Running this in parallel reaches speeds of extraction of around 15 files per second. Whereas, the top speed would be 2 files per second without.

Additionally, a separate file adds environmental information into the dataset. The decision to create an independent file and a distinct dataset is because this addition diverges from the original research question. Nonetheless, incorporating this information is valuable for real-world applications. Therefore, it was decided to include this data in the final report but to differentiate between classification using image features or with environmental data added. The procedure begins by accessing the GeoTIFF files that contain the environmental data. These data are represented as arrays, with each point corresponding to a $1km^2$ tile. The affine transformation is reversed to find the value based on East and North coordinates.

4.3. City Classification. Before being able to perform classification, some modifications must be applied to the model. First, the city locations are turned into categorical data. Then, single number features are isolated and scaled. Distribution features must first, be turned into an array, that array is then scaled. All of our inputs and outputs are then split into a Train and Test dataset.

With our inputs and outputs created and split we may create the model. The

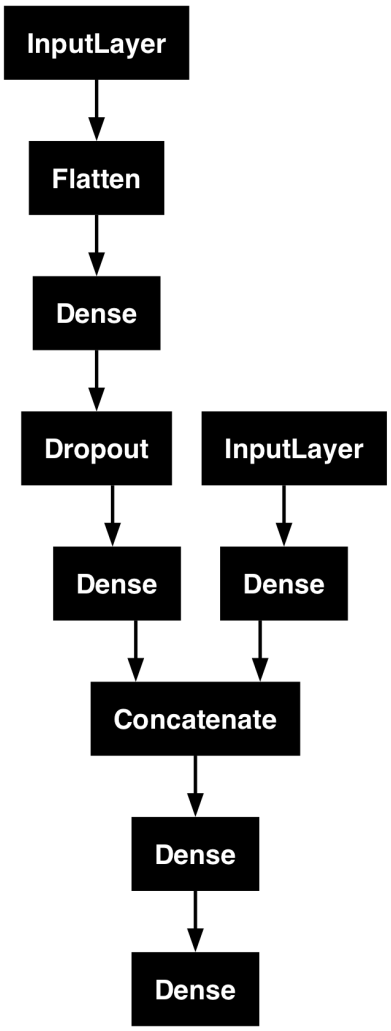

FIG. 6. *Multiple input model plot*

model plotted in Figure 6 and Figure 7 summarizes the model

5. Results. The model is fitted with a Cross Entropy loss function and accuracy is taken as the metric. For each model, the accuracy and the confusion matrix on the test dataset is calculated.

5.1. 10 biggest cities in Switzerland. When taking the 10 biggest cities in Switzerland the results could be better. Indeed, with an accuracy of 50%, the model seems to suffer from the imbalance of observations. The confusion matrix in Figure 8 show difficulty in distinguishing between Winterthur and Zürich, two cities in the same canton and very close to each other. Bern is also often confused with Zürich. Geneva and Biel both have a very low accuracy. Lugano is very well classified, this is due to the city being far away from the rest.

the addition of supplementary information helps raise the accuracy to a satisfying 70% and the confusion matrix in Figure 9 shows more precision.

Layer (type)	Output Shape	Param #	Connected to
Image (InputLayer)	(None, 256, 4)	0	–
flatten (Flatten)	(None, 1024)	0	Image[0][0]
dense_1 (Dense)	(None, 128)	131,200	flatten[0][0]
Numeric (InputLayer)	(None, 9)	0	–
dropout (Dropout)	(None, 128)	0	dense_1[0][0]
dense (Dense)	(None, 64)	640	Numeric[0][0]
dense_2 (Dense)	(None, 128)	16,512	dropout[0][0]
concatenate (Concatenate)	(None, 192)	0	dense[0][0], dense_2[0][0]
dense_3 (Dense)	(None, 64)	12,352	concatenate[0][0]
dense_4 (Dense)	(None, 10)	650	dense_3[0][0]

FIG. 7. *Summary of the model*

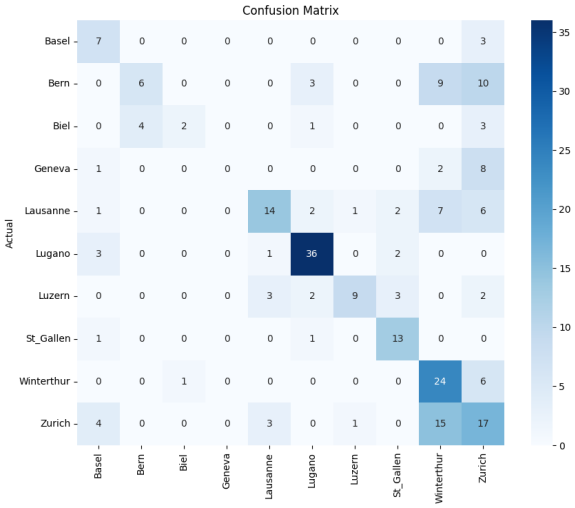

FIG. 8. *Confusion matrix 10 biggest cities*

5.2. Balanced dataset. The results here are around 50% this is again somewhat disappointing. The confusion matrix in Figure 10. Shows that the main issues are again between Winterthur and Zürich, but also Schwyz and Glarus. This is not surprising as both pairs are quite close geographically. The confusion between Chur and Lugano is also quite surprising, this may be due to the location of both in a mountainous area.

With the addition of environmental information, the accuracy climbs to around 80%. This result is quite satisfying and supported by a confusion matrix in Figure 11 boasting a high precision.

6. Improvement areas. While overall the results can be considered as satisfying, some improvements could be made. In terms of the download of the data, a list of links for the the whole of Switzerland. These links contain in their URL the location of the top left corner of the tile. Using a dataset containing boundaries of each city named swissBOUNDARIES3D, in which every city is registered as a Shape item. it would be possible to simply create a process where the user would simply enter a list of cities and using these shapes and tile locations, every tile contained inside a city could be downloaded. This will not be explored in this paper, but may

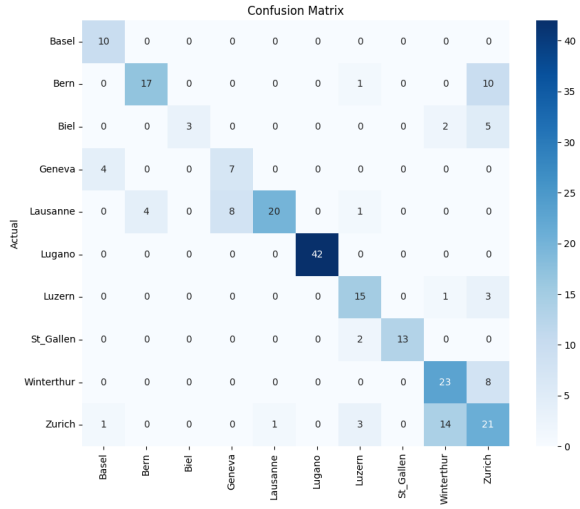

FIG. 9. *Confusion matrix 10 biggest cities with env. info*

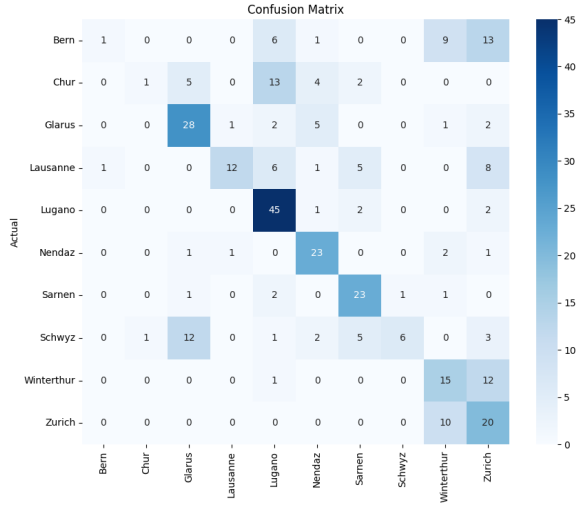

FIG. 10. *Confusion matrix balanced dataset*

be added later on. In term of the features extracted, they provide an accurate mix of useful features while also staying low in term of storage usage. Further features were attempted but eventually discarded. This is the case of edge detection, edge detection assumes that edges in an image will reflect in a sharp change between two adjacent pixels. using this for aerial imagery accurately recognizes features such as buildings, roads, rivers or mountains. Lastly, guessing the coordinates instead of the city could be attempted. Being able to guess coordinates would be a step further unlocking many possibilities. Anyone is welcome to build upon this project and add new functionalities, the repository containing the code can be found using

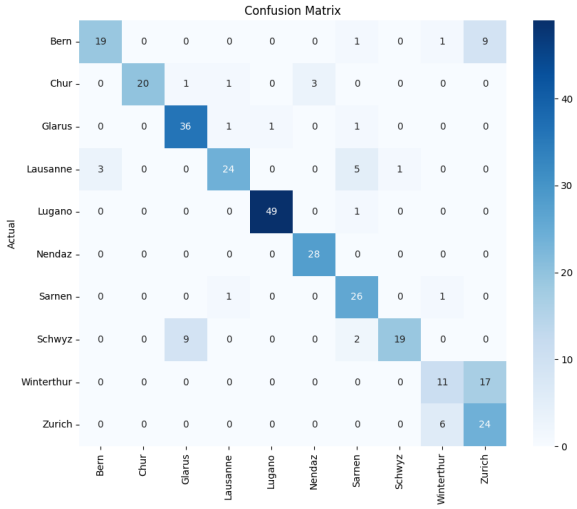

FIG. 11. *Confusion matrix balanced dataset with env. info*

https://github.com/Elwin-Freudiger/Swisscities.

7. Conclusion. In conclusion, our results demonstrates the possibility to classify a city using only features extracted from an aerial image by training a deep neural network model, it also shows that environmental features are of great help for accuracy. This highlights the need of contextual data for location classification.

Many applications could stem from this. The detection of natural features, classification between urban and rural area could be implemented.

The rise in the usage of drones also makes this research more relevant.

Overall, this research shows that great results can be obtained with very few features, computational power and an easy to implement deep learning model.

Appendix A. Helper Tools. Chat-GPT was used in this project for bug-fixing and to correct grammar in some parts of the text.

Acknowledgments. The author of this paper would like to thank the *Federal Office of Topography swisstopo* for making various datasets used through this project available.

REFERENCES

[1] Y.-C. CHENG AND S.-Y. CHEN, *Image classification using color, texture and regions*, (2001). Received 14 October 2001.

[2] R. M. HARALICK, K. SHANMUGAM, AND I. DINSTEIN, *Textural features for image classification*, IEEE Transactions on Systems, Man, and Cybernetics, SMC-3 (1973), pp. 610–621, https://doi.org/10.1109/TSMC.1973.4309314.