Cartography M.Sc.

Master thesis

Colouring and interactive visualization of historical Earth observation data

Yiwei Wang

2022

Colouring and interactive visualization of historical Earth observation data

submitted for the academic degree of Master of Science (M.Sc.)
conducted at the Institute of Cartography and the Institute of Photogrammetry and Remote Sensing, Department of Geosciences
Technical University of Dresden

Author:	Yiwei Wang
Study course:	Cartography M.Sc.
Supervisors:	JProf. Dr. Matthias Forkel (TUD)
	Mr. Mathias Gröbe M.Sc. (TUD)
	Mr. Lucas Kugler M.Sc. (TUD)
Reviewer:	Olesia Ignateva (TUW)

Chair of the Thesis
Assessment Board: Prof. Dr.-Ing.habil. Dirk Burghardt (TUD)

Date of submission: 08.09.2022

Statement of Authorship

Herewith I declare that I am the sole author of the submitted Master's thesis entitled:

"Colouring and interactive visualization of historical Earth observation data"

I have fully referenced the ideas and work of others, whether published or unpublished. Literal or analogous citations are clearly marked as such.

Dresden, 08.09.2022 Yiwei Wang

Acknowledgement

First and foremost, I would like to sincerely thank all my supervisors, JProf. Dr. Matthias Forkel, Mr. Mathias Gröbe and Mr. Lucas Kugler, for kindly helping and supporting me through my research and providing me with valuable suggestions and insights into my study when I encounter difficulties.

I would like to express my gratitude to all my fellow Cartography students. It has been a wonderful two years of study. We have had a great time together in Munich, Vienna and Dresden. This will forever be a precious memory of mine.

I also sincerely appreciate all participants of the user study for taking the time to help me. Without your kind help, this study would not have been accomplished.

I would like to thank Jason Antic and all other contributors to the amazing DeOldify model. It is your great work that has made my research possible. And I would also like to thank GitHub and Sci-hub for sharing knowledge and opening the door of science to everyone.

Finally, my deepest love goes to my family and my friends, who would always support me through the difficult times during my master study in a foreign country. I shall never forget all your encouragement and loving words.

Abstract

In the 1960s, the US launched a series of satellite projects aimed at building a reconnaissance system. The CORONA project is in operation between 1960 and 1972. During the operation, a large number of satellite images are taken. These generated images are called CORONA images. Until 1995, these images remained classified. Numerous researchers have exploited CORONA images and performed studies on changes on the earth's surface. This study adapts the existing DeOldify model, which is dedicated to colourizing images to specifically colourize CORONA satellite images through retraining the model on the CORONA image dataset. The CORONA image dataset consists of greyscale CORONA images and colour reference images covering the same region and having little or no change in the image content. A generative adversarial network is used for building this model. A U-Net-based generator network and a binary-classifier-based critic network train in an alternative approach. To improve the training efficiency, transfer learning and NoGAN training techniques are implemented. The generated images are evaluated both quantitatively, using RMSE and PSNR, and qualitatively, using a user study asking the participant if the displayed colour image looks natural. Results show that generated images with good performance on RMSE and PSNR do not necessarily have plausible colours. Compared with the original DeOldify model, the retained model can produce images with more natural and plausible colours, achieving a result of 70.1% of the participants thinking they are real, although these images have poorer reconstructing quality. To visualize the results, a web mapping application is developed with Geoserver and Leaflet. The server stores data and publish it through WMTS or WFS. Cache service and image pyramid are used to accelerate the response on the client side. The web map client provides an interactive map interface for the users to control the displayed layers and download the selected layer in a self-defined ROI through WPS.

Keywords: *Generative adversarial network, user study, web mapping application*

Contents

Figures

1 Introduction

1.1 Motivation and problem statement

In the 1960s, the US launched a series of satellite projects aimed at building a reconnaissance system (Peebles, 1997). The CORONA project is the one that was operating between 1960 and 1972. The purpose of the CORONA project was to provide information and produce maps for the US intelligence agencies. During the CORONA project, the Keyhole (KH) designators were used to name different commissions (Earth Resources Observation and Science (EROS) Center, 2018). This includes KH-1 to KH-3, which provide images from a single panoramic camera at a ground resolution of 7.6 meters. KH-4 provides images taken from two panoramic cameras at the same ground resolution. Later, KH-4A and KH-4B commissions could provide images with a better ground resolution taken from two panoramic cameras, at 2.7 meters and 1.8 meters respectively (Earth Resources Observation and Science (EROS) Center, 2008). These so-called CORONA images are available at a high spatial resolution but provide only panchromatic (grey-scale) information. Additionally, the images were taken with different camera systems and under different observation geometries, which causes various radiometric properties and diverse visual appearances of the images (Oder et al., 2013).

In 1995, these images were declassified and opened for public access (Galiatsatos et al., 2005). Since then, the Corona images have become very valuable for the assessment of historical changes in land cover and land use: Li et al. (2003) investigate the reasonable causes and consequences for land-use and land-cover changes in the region of Tarim Basin using a CORONA panchromatic image taken in 1964 and a Landsat ETM image in 2000, Andersen (2006) studies the process of long term deforestation in the Eastern Desert of Egypt with CORONA images from 1965 and field data from 2003, Cetin (2009) examines the coastal regional trends in urbanization processes in western part of Istanbul using the CORONA and LANDSAT TM satellite images of 1963, 1987 and 2005 with a post-classification comparison method, Shahtahmassebi et al. (2017) propose a method to derive texture from CORONA images to study the relationships with spectral vegetation indices and texture variables derived from Landsat MSS to reconstruct historical continuous land cover type and complexity, Jabs-Sobocińska et al. (2021) use CORONA satellite images to map agricultural lands which converted into the forest as a result of widespread land abandonment due to Post-WWII displacements in the Polish Carpathians.

However, due to their grey scale, one distinct disadvantage of the declassified CORONA images is that they only contain limited radiometric information. Colourizing greyscale images enables people to obtain abundant visual information from the colourized images, which helps distinguish and recognize image content (Todo et al., 2019). This study tries to find a way to colourize CORONA images and achieve a natural-looking result. Furthermore, the results should be visualized in an interactive web interface, which allows users to browse and possibly download interested data.

1.2 Research objectives and research questions

The overall research objective is to colourize the grey-scale CORONA images taken from the 1960s to 1970s in the region of Saxony and visualize the results. Specifically, this study is divided into the following three research objectives:

1. To propose a deep learning-based method to colourize grey-scale CORONA images and achieve a natural-looking result.
2. To propose quantitative and qualitative measurements to evaluate the quality of the colourized images.
3. To design a web-based interactive interface to display the obtained results and allow users to browse and possibly download the interested dataset.

To achieve the desired results, this study aims to answer the following research questions:

a) What are existing deep learning-based methods to colourize images and how to adapt them to apply to CORONA satellite images?
b) Which quantitative measurements can be used to evaluate the similarities between artificially colourized CORONA images and ground truth images?
c) How to design a user study to evaluate the plausibility of artificially colourized CORONA images?
d) How would humans perceive a colourized image that has a good performance in quantitative measurement as natural looking?
e) How to build the interface of the webpage and display large satellite images?
f) What are suitable workflow and implementations to provide custom extraction of displayed data?

1.3 Innovations

Much research has been done in the field of image colourization using various methods. However, many of them focus on general images like landscape, portraits and simple

objectives. This study aims to use Generative Adversarial Networks (GANs), a deep learning-based method, to colourize grey-scale satellite images from the CORONA program. To evaluate how natural-looking the coloured images are, this study uses both quantitative and qualitative measurements and compares the performances in these two aspects.

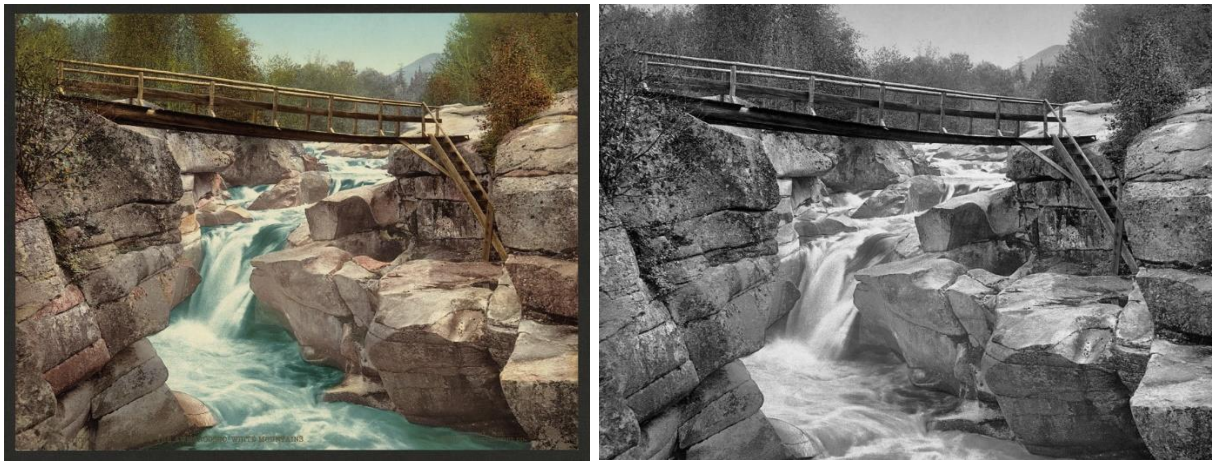
1.4 Structure of the thesis

The structure of the thesis is as the following: chapter 2 introduces the previous studies on image colourization problems and web mapping techniques. Chapter 3 describes the data and method used in this study. Chapter 4 contains the experiments conducted in the study and the results and analysis. Chapter 5 provides the conclusion and possible improvements and future work of this study.

2 Background

2.1 Image colourization

Image colourization is the process of colouring monochrome images and has been widely used in photo processing and scientific illustration (Luan et al., 2007). It is not a new subject. The earliest attempt to colourize black and white photos can be traced back to the 19th century when the daguerreotype became the first successful commercial photographic process and people hand-coloured photos with the help of colour pigments mixed with gum arabic or quicker-drying mixtures containing alcohol (Weitz, 2020). Later, the Swiss-invented method Photochrom also gained commercial success in the hand-colouring of black and white photos (Gable, 2009). This involves using lithographic stones and light-sensitive chemicals. Postcards made by lithographers were very popular among tourists at that time. Both two methods require heavy handcraft and are time-consuming work due to the complex process of hand-colouring. But the produced work is generally muted, lacks adequate shading and appears unrealistic. Another common characteristic of these photos is that they are often coloured by people who have never actually seen the pictured scene in actual colours. Subsequently, these colours only reflect the aesthetic views of the creators but do not necessarily resemble the true appearance.



(a) (b)

Figure 1: Example of Photochrom colour photo (a) and corresponding black and white photo (b) (Gable, 2009)

In the 1970s, the former NASA engineer Wilson Markle first introduced the term colourization, describing it as the process whereby black-and-white motion-picture material is transformed into colour videotape. With the help of the development of the computer, colourization also transformed itself from pure handcraft to a computer-aided process.

But human intervention is still required in Markle (1984) method. An art director is needed and must be familiar with how to segment the scene, special effects and television signal transmission, as this person needs to provide detailed information for the proper selection of colours. Prior research is often required, and personal experience must be provided when lacking such information.

Due to the fast development of computer systems and later the introduction of neural networks, image colourization has changed from simple but time-consuming human intervention-based methods to complex but highly automated deep learning-based methods (Žeger et al., 2021). Before deep learning became popular and developed, image colourization in the computer domain was mainly divided into two categories: scribble-based and example-based. As neural networks gained popularity, deep learning-based methods significantly outperform previous methods.

2.1.1 Scribble-based methods

Scribble-based methods still require human intervention. Usually, that means users have to draw a set of colour scribbles at specific positions in the black and white image, and then the algorithm would decide how the colours propagate and thus colourize the whole image. Similarities can be found in earlier hand colouring techniques. The image is segmented into regions according to the context, and then proper colours are chosen and applied to each region. But without the aid of the computer, the whole process is made by handcraft and thus is highly time-consuming and with low efficiency. Levin et al. (2004) firstly proposed a simple but effective scribble-based method. The colour scribbles provided by the users propagate based on the simple premise that nearby pixels with similar grey values should have similar colours, i.e., intensity and colour should match a linear function. Images are first converted from *RGB* colour space to *YUV* colour space, where *Y* indicates luminance intensity, and *U* and *V* are chrominance channels representing colours. The idea is to solve a weighted equation problem that pixels in the neighbouring region should have minimal quadratic chrominance differences sum, where the weight increases as the intensity value difference between two pixels decrease. The choosing of weights is sometimes also referred to as affinity functions.

(a) (b) (c)

Figure 2: User-provided scribbles image (a), colourized image (b) and reference image (c) (Levin et al., 2004)

Based on Levin et al.'s work, Huang et al. (2005) proposed an improved method to eliminate the bleeding effect occurring around the object boundary during the colourization process. An adaptive edge detection scheme, which is a four-step method based on a Sobel kernel, is used to extract reliable edge information to form a clear edge map before the propagation of the colour scribbles. The weighting function is also adapted to ensure that pixels with similar intensity values also have similar chrominance values and the changes should be proportional. Yatziv and Sapiro (2006) used the geodesic distance to evaluate the proximity between two pixels in the luminance channel, thus assigning them chrominance values accordingly. Differently from the previously mentioned two methods, the image is converted to the *YCrCb* colour space rather than *YUV* colour space. Qu et al. (2006) proposed a new method for image segmentation which can handle images with little or no continuity of grey levels but is rich in the continuity of patterns. Based on the segmentation, colour scribbles stop propagating at the boundary where the pattern exhibits abrupt change. Furthermore, non-adjacent regions with similar patterns can also be coloured the same. While this method works well on Manga colourization, it doesn't have a good performance on natural images rich in inhomogeneous texture patterns. To overcome this shortcoming, Luan et al. (2007) proposed a two-step colourization system consisting of colour labelling and colour mapping. Users only need to provide a small number of colour strokes to each group of regions that roughly share similar colours. In each group of regions, a more precise colour is assigned to a few pixels with significant intensity values for fine-tuning. To further reduce the required amount of user inputs, An and Pellacini (2008) and Xu et al. (2009) proposed affinity-based edit propagation to ensure that the least amount of user input could propagate to the whole images based on the policy that spatially-close regions should have a similar appearance.

The distinct disadvantage of scribble-based methods is that user intervention is required and heavily relied on. To colourize a black and white image without its corresponding true colour reference, the user must have adequate aesthetic knowledge and

experience to produce a plausible result. Besides, the number of colour strokes needed for colourization could be significant and thus leading to low efficiency and a high cost of time. However, the upside is that few or no reference images need to be provided. Also, the calculation of the propagation algorithm is relatively simple, mostly involving the solution of a minimization problem of a weighting objective function.

2.1.2 Example-based methods

Different from scribble-based methods, which require human intervention to colourize the image, example-based methods take advantage of reference colour images and perform a colour transfer to black and white images or recolour colour images. No user input or only a small amount of user input is required, thus achieving automation of the whole process.

Colour transfer method is first used for recolouring colour images. Reinhard et al. (2001) proposed a simple statistical-based method to achieve image-to-image colour transfer. Colour images in classic RGB colour space are converted to $l\alpha\beta$ colour space (Ruderman et al., 1998) where the coordinate axes of the three colour channels are orthogonal to each other, thus creating few or no correlations between each other. This conversion simplifies colour modification as each channel is now separate and independent. Then, to transfer colour from one image to another is to simply modify the data distribution of target image in $l\alpha\beta$ colour space to match the one of source image. Welsh et al. (2002) extended this method to colourize grey-scale images. Different from colour images, grey scale images only contain one luminance channel. Colour information matching thus is downgraded to intensity information matching. Luminance remapping is first performed to linearly shift and scale the luminance histogram of the source image to fit the histogram of the target image. Each pixel in the source colour image is then matched to another pixel in the target grey image according to L_2 distance in luminance values in a local neighbourhood. Both global and regional differences are considered for more precise matching. Colours are thus transferred from the source image to the target image.

(a) (b) (c)

Figure 3: Colours are transferred from source image (a) to the target image (b) (Welsh et al., 2002)

The aforementioned methods often produce colourized images with hard and clear boundaries between different regions, creating an unnatural look. Instead of assigning each pixel in the source image to only one pixel in the target image, Tai et al. (2005) matched pixels according to the probabilistic segmentation by a new expectation-maximization scheme. Every pixel is given a probability that it belongs to a certain region rather than directly matched with another pixel. Pixels in the centre of the region tend to have a higher probability of falling into this single region, while border pixels have a more uniform probability distribution. Inter-region smoothness can be achieved, and thus fewer unnatural colours would appear around the boundary of a region. A supervised learning technique is also applied to decide which region one pixel belongs to (Irony et al., 2005). Due to the concern that different regions could have pixels with the same luminance value and similar neighbourhood statistics, simply assigning pixels by probability statistics is insufficiently accurate. Thus, a feature space where similar pixels in different regions should be discriminated is constructed after automatic image segmentation or user's manual marking based on local frequency analysis in the luminance channel. A mapping function between the local pixel neighbourhood and pixels in the feature space is established, and pixels are assigned to a specific region by voting in both image space and feature space and then colourized.

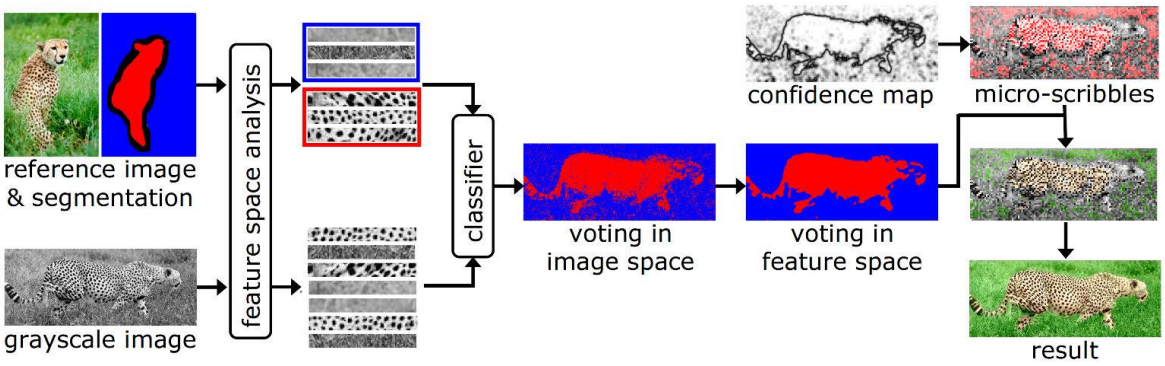


Figure 4: Structure of Irony et al. (2005) method

Besides Ironi's method, feature extraction and matching are also widely used in other example-based colourization methods. These methods generally construct a feature space by computing multi-dimensional feature vectors to precisely or ambiguously map pixels between source and target images. Charpiat et al. (2008) computed Speeded Up Robust Features (SURF) (Bay et al., 2006) descriptors at multiple image scales followed by principal component analysis to reduce the number of feature vectors. Gupta et al. (2012) incorporated highly discriminative SURF and Gabor features to

find matching pixels between source and target images. Feature extraction is at super-pixel resolution to speed up the later colourization process and maintain stronger spatial coherency. Wu et al. (2013) applied image segmentation to separate background and fore-ground regions. Multiple statistical and spatial features are calculated on each super-pixel segmented by Mean Shift (Comaniciu & Meer, 2002) in each region to construct a soft many-to-many mapping between super-pixels.

To overcome the difficulties in finding a suitable colour reference image, Liu et al. (2008) proposed a method to use images from web search results to colourize a target image and eliminate differences in illuminance conditions at the same time. However, this method is only applicable to well-known landmarks where images can be easily found on the internet. Chia et al. (2011) expanded this method to general images by using user-provided semantic text labels to search for similar images on the internet and filtering them to get suitable reference images. Target images firstly need to be segmented to the background and foreground objects and given text labels by users.

One distinct disadvantage of the example-based methods is that users must provide a colour reference image. In most cases, this reference image is expected to have similar patterns and visual consistency as the image to be colourized. Although this problem is already properly addressed in Liu's and Chia's study. Different extents of user intervention are still required. But compared with scribble-based methods, example-based methods require much less amount of user interaction, thus leading to higher efficiency. The whole process of colourization is more automatic, and image segmentation and feature extractions are highly involved.

2.1.3 Deep learning-based methods

Example-based methods can already achieve a highly automated process for image colourization. But the output heavily relies on finding a suitable reference colour image, which can be difficult sometimes. The development of deep learning techniques has enabled the usage of artificial neural networks, which could be trained with a large number of training data. For an image colourization process, this means that the problem of finding a suitable reference image can be properly addressed by using a large number of source images and letting the neural network find the best matching part.

Cheng et al. (2015) first applied deep neural networks to an image colourization problem and solved it as a regression problem. The whole process is divided into two steps: the training step and the testing step. In the training step, the neural network automatically learns a mapping function $\mathcal{F}$ between a greyscale image G and a colour image C

given a set of such image pairs. To learn the function parameters Θ, the following least squares minimization problem is solved:

$$\underset{\Theta \subseteq \Upsilon}{\operatorname{argmin}} \sum_{p=1}^{n} \left\| \mathcal{F}(\Theta, x_p) - c_p \right\|^2 \tag{1}$$

where x_p is the input feature descriptor at the pixel p, c_p are the corresponding chrominance values in the reference colour images, n is the total number of pixels, and Υ is the function parameter space. Similar to previously mentioned example-based methods, to minimize the correlations among colour axes, the YUV image space is used, where Y is the luminance channel, and U and V are chrominance channels containing colour information. The neural network takes a feature descriptor calculated at pixel p in the greyscale image and outputs the predicted chrominance values. Input features consist of a three-level feature collection: a low-level patch feature representing the greyscale values in the local 7x7 pixel neighbourhood, a middle-level DAISY feature (Tola et al., 2008) and a high-level semantic feature. The low-level and middle-level features contain intensity and geometry information and work well on low-texture regions and complex regions, respectively, while the high-level feature indicates the semantic information for each pixel and can reduce ambiguities while matching. A post-process of joint bilateral filtering is used to reduce the artefacts in the generated chrominance channels.

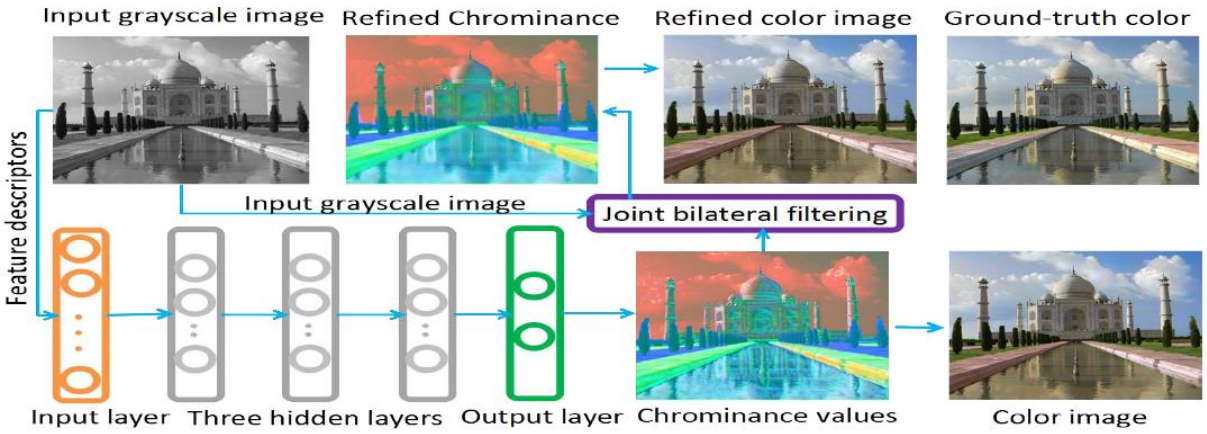


Figure 5: Overview of the proposed colourization method and the architecture of the adopted deep neural network (Cheng et al., 2015)

The architecture of Cheng's model is a simple-designed deep neural network, which does not keep the spatial information while inputting the image. Instead, the input is a set of calculated feature vectors. Convolutional Neural Networks (CNNs) have been

widely used in image processing problems due to their ability to maintain spatial information in the images. Thus, they are also applicable in solving image colourization problems. A typical CNN could have the following layers: an *input layer* which takes an image of one channel (greyscale image) or three channels (RGB image or in other colour space) as the input of the neural network, and a *conv layer* which consists of a set of filters and outputs the convolution results, a *Rectified Linear Unit (ReLU) layer* which acts as the activation function, a *pooling layer* which performs down sampling along the spatial dimensions, a *Fully-Connected (FC) layer* which computes the final classification results.

One of the most well-known CNNs used for image colourization was proposed by Zhang et al. (2016). Similar to Cheng's study, the input image is first converted to CIE Lab colour space where L is the luminance channel, and a and b are the chrominance channels. The network takes the L channel as input and predicts the two chrominance channels. As shown in Figure 6, each block contains 2 or 3 repeated *conv* and *ReLU* layers followed by a *BatchNorm* layer. Instead of using a *pooling* layer to reduce the spatial size, a stride 2 *conv* layer is applied at the end of each block for this task. An L_2-loss function is commonly used in CNNs. It calculates the Euclidean distance between the ground truth and predicted results in the chosen colour space. This is simple and intuitive. But in the colourization problem, it cannot handle the multimodality of colour distribution, and thus usually leads to desaturated results because it encourages the network to produce a mixture of multiple possible colour pairs to get the minimal loss value. To solve this problem, colourization is treated as a multinominal classification problem with 313 ab value pairs according to empirical experience. A multinominal cross-entropy loss function of predicted colour pairs and ground truth colours is used. To reduce the strong influence of low-value ab pairs in natural images, the ground truth value is first transformed to a vector with a soft-encoding scheme and then rebalanced based on colour rarity.

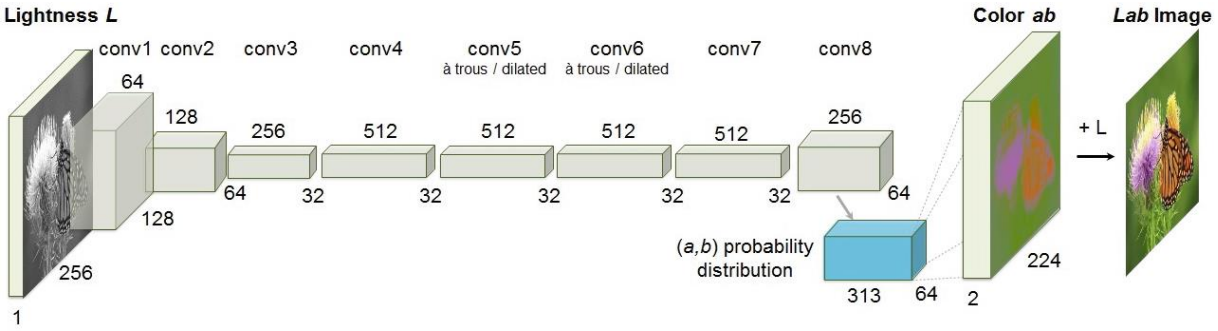


Figure 6: Architecture of Zhang et al. (2016) proposed network

Image features can also be learned from multiple paths and later combined for better performance. Iizuka et al. (2016) proposed a CNN-based method without pre-processing or post-processing and can achieve an end-to-end performance. Different from Zhang's network which only considers local image features, Iizuka's method applies low-level, mid-level and global level features and combines local and global features with a fusion layer as shown in Figure 7.

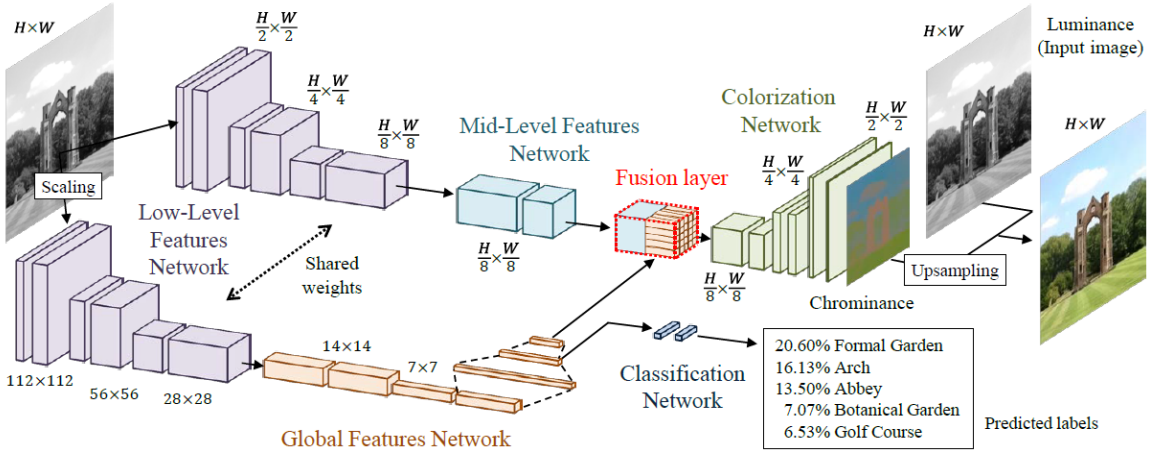


Figure 7: Architecture of Iizuka et al. (2016) proposed network

The network works on images in CIE Lab colour space and takes the intensity channel as input and outputs two chrominance channels. Global features are computed not only on the input image but on all images in the training dataset. It is possible to also achieve a style transfer between different images. Each image is tagged with a class label to guide the extraction of global features. Thus, a small classification network is also trained. The global loss function of the whole network is the Mean Square Error from the colourization network and the cross-entropy loss from the classification network. Backpropagation and adaptive learning rate are also applied to optimize the learning of the network. Larsson et al. (2016) used a hypercolumn to preserve a per-pixel descriptor in spatially localized multilayer slices in a VGG-16-structure-based CNN. The colourization system takes this hypercolumn as input and predicts the lightness and chrominance distributions for each pixel. The system also predicts the probability over a set of colour distributions and uses KL divergence to measure the distance between the predicted distribution and ground truth colour distribution.

Another widely used network type in solving image colourization problems is Generative Adversarial Networks (GANs) which were first introduced by Goodfellow et al. (2014) to produce images from random noise input. The basic architecture of a typical GAN contains two separate networks: a Generator (G) which generates fake data from the input, and a Critic (C) or Discriminator (D) which discriminates between fake data

generated by G and ground truth. The competition between the two networks could produce vivid and plausible results in solving image-related problems. In image colourization, conditional Generative Adversarial Networks (cGANs) are more often used because they also take an image as input.

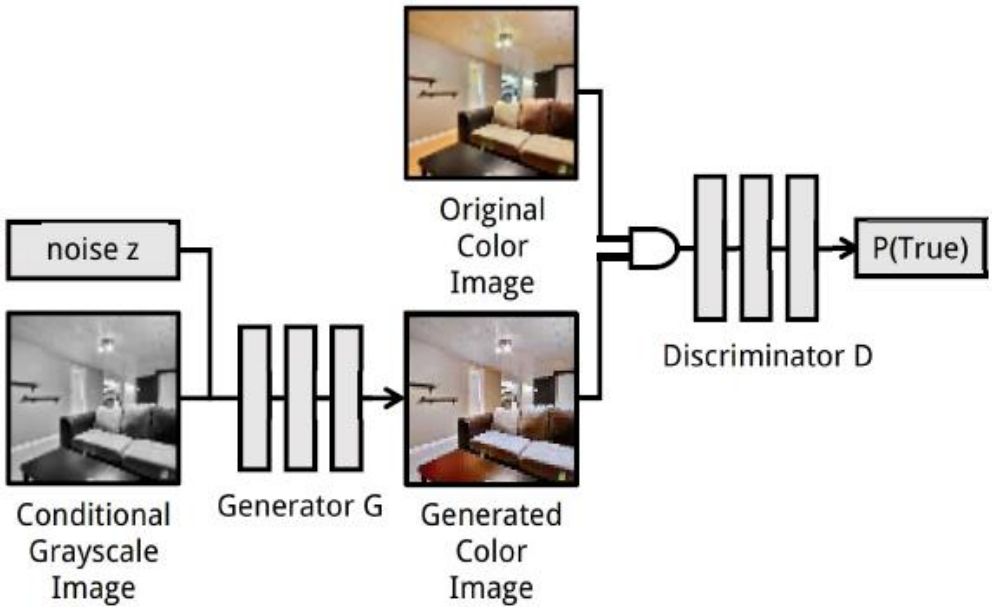


Figure 8: An illustration of cGAN architecture (Cao et al., 2017)

For the generator to output an image, one of the chosen architectures is in an auto encoder style as in Isola et al. (2017). The encoder performs a series of downsampling operations until it hits a bottleneck part, and then the decoder performs a series of upsampling operations to reconstruct the original image size. Skip connection is added to retrieve low-level features from images in early layers so that the produced results can largely retain the details of the image. Wu et al. (2021) also implemented skip connection in the generator in the style of a Residual Neural Network (ResNet) (He et al., 2016). To avoid overfitting due to the increase of layer depth, a multi-scale convolution with different kernel sizes in one convolution layer is applied. Cao et al. (2017) applied skip connection without changing the spatial size of the input image in the whole generator. Only convolution of stride 1 will be performed. For each *conv* layer, it is concatenated with the original input image to preserve important feature information. The introduction of skip connection could greatly improve the visual quality of produced image results because texture details and features can be well maintained and reconstructed.

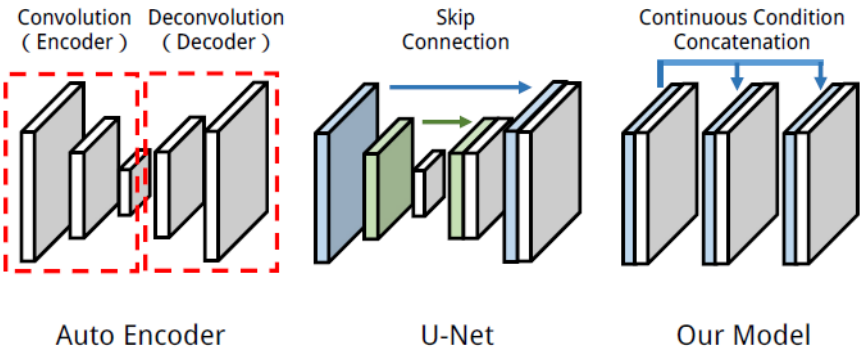


Figure 9: An illustration of auto-encoder, U-Net and Cao et al. (2017) method

2.2 Web mapping

2.2.1 History and development

Web mapping is the process of designing, implementing, generating, and delivering maps on the World Wide Web (Neumann, 2012). It usually involves geodata and its visualization (in the form of a map) on the web and the technologies it implements. Since shortly after the creation of the internet and web, web mapping technologies have been developing at a very fast pace (Veenendaal et al., 2017). A typical web map system consists of two basic components: a web interface displaying the map and a server providing the data. The first published web-based map is Xerox PARC Map Viewer (Putz, 1994), which can display a static map as an image on the web and generates a user-input-defined map area. The realization behind this simply designed web mapping interface is the implementation of Hypertext Transfer Protocol (HTTP) and Hypertext Markup Language (HTML), and the server is running on a workstation using a custom server module written in *Perl* script language.

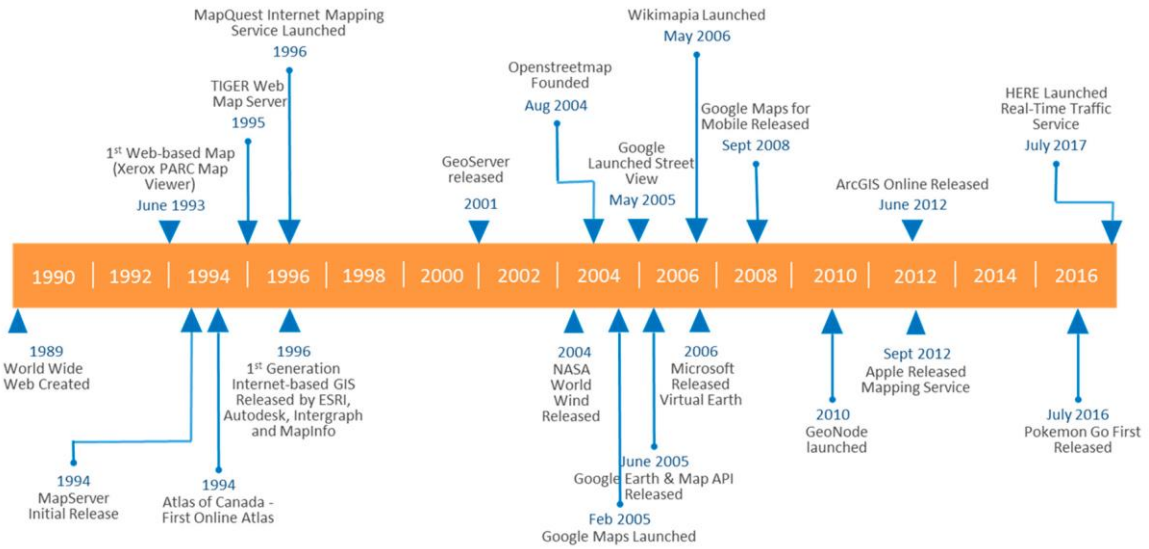


Figure 10: Timeline of some significant web mapping events (Veenendaal et al., 2017)

Static web maps can be regarded as the first generation of web mapping applications. It is fully based on the Web 1.0 technology of HTTP and HTML. A published map consists of a series of images and the views of the map can be changed to focus on different geographic regions by simple clicks on hyperlinks. It essentially works as an interface for the user to retrieve information. But the client-server architecture has already been established and will continue to evolve in the following decades. As web technologies evolved from Web 1.0 to Web 4.0 (Choudhury, 2014), the web mapping application has also gained development. A dynamic web map was introduced to satisfy the need that multiple users to share the map and modify or customize it accordingly. Dynamic web maps take advantage of Dynamic HTML, Common Gateway Interface (CGI) to build the client side, and Java applets and servlets, plugins and ActiveX technologies for the server side. Maps are dynamically generated on the server according to user preferences and choices (Peng & Tsou, 2003). Instead of displaying a set of pre-generated maps to the users in static web maps, dynamic web maps enable users to retrieve information based on their requests and needs. A certain degree of user interaction such as manipulating map view has become an important feature in web mapping applications.

The increased user needs have also put a challenging task on web mapping servers and thus new development has been achieved to improve the performance of map servers. Service-oriented architecture (SOA) was introduced to facilitate the development, publishing and implementation of web map servers (Sahin & Gumusay, 2008). Components-based structure enables each component to be reusable and greatly improves efficiency. Geoserver built upon Java language is an example of such implementation (Barik et al., 2011). Application programming interfaces (APIs) take the advantage of well-designed map servers and enable users to use services from different online servers in an organized, easy-to-read way to build a shareable client interface (Veenendaal et al., 2017). Among many APIs created for different purposes, some well-known APIs dedicated to web mapping include OpenLayers, Leaflet, ArcGIS API for JavaScript, and services from online map providers like Google Maps and Bing Maps. Most of these APIs are built upon JavaScript and HTML and provide users with a large number of GIS functionalities over data control, management, and display.

Figure 11: MapQuest is one of the first examples to incorporate web mapping services

The introduction of asynchronous JavaScript and XML (AJAX) further improved the efficiency of the web mapping application. Instead of waiting for the response from the server, users can interact with the web map interface simultaneously (Geospatial World, 2009). Requests submitted are no longer handled in a single thread. The delivery of results to the users is greatly accelerated.

The data source for web mapping applications has benefited from Volunteered Geographic Information (VGI) since Web 2.0 enabled more users to participate in web mapping. OpenStreetMap was founded in 2004 and has since been a rich geodata source. Users from all over the world can contribute to the data collection. A large amount of geodata is published on the platform and can be retrieved under an open-source license. Combined with different types of data, VGI can also be used in 2.5D or 3D web mapping applications with the help of supporting JavaScript libraries. In addition, the platform of web mapping is expanded from desktop web browsers to mobile devices. In return, the acquisition of VGI becomes easier as mobile devices are portable. With highly integrated APIs, reusable components for servers, and a rich source of open geodata, the development of web mapping applications has been largely facilitated.

2.2.2 OGC standards

To standardize the various mapping services, a set of standards and specifications have been developed by the Open Geospatial Consortium (OGC). These include data visualization services like Web Map Service (WMS), data publishing services like Web Feature Service (WFS) for vector data and Web Coverage Service (WCS) and access to geospatial processing services Web Processing Service (WPS).

WMS provides an easy way to request georeferenced static map images from various geodatabases by simply sending an HTTP request along with parameters. These parameters can specify the requested data source, projection, extent, size, output format (like PNG or JEPG), etc.

WFS and WCS provide direct access to vector and raster data respectively from geodatabases. Similar to WMS, the implementation is also achieved through an HTTP request with relevant parameters. The requested data can also be displayed on a web map client like what a WMS request can achieve. But the data can be accessed in their original formats or other specified formats and directly downloaded. Furthermore, WFS enables users to create, modify or delete data in the requested geodatabase, and retrieve data or its attribute information according to defined constraints. WCS can also handle multidimensional data. Raster data can also be queries based on user-defined constraints through WCS.

WPS enables users to request geospatial processing services published on the server from the client side. It regulates how the input and output are handled and allows the user to query the execution progress of the process.

2.2.3 Tiling and caching services

WMS facilitates the users to request data as static map images of any size. But this reduces the potential to cache images and results in regenerating images whenever the request is received on the server side (García et al., 2012). Map view changes like pan or zoom could also result in a re-rendering of the whole map and thus requesting again. To address this problem, tiling service and cache services are proposed. A common way to achieve this is to cut the whole map into smaller tiles. These image tiles are stored and pre-rendered in the data server, and can quickly respond to the user's request. When requesting is needed again due to the map view changing, only the relevant image tiles are sent to the client side (Sample & Ioup, 2010). The Open Source Geospatial Foundation (OSGeo) proposed WMS Tile Caching (WMS-C) standards and the OGC developed the Web Map Tile Service (WMTS) standards respectively.

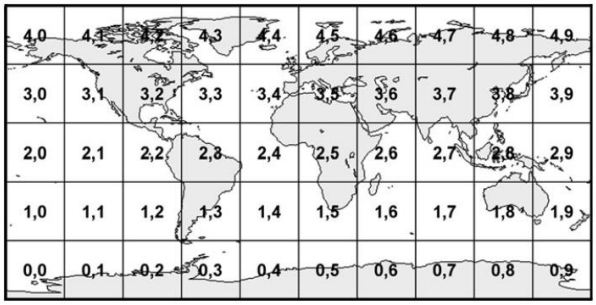


Figure 12: An example of tiling a map (Sample & Ioup, 2010)

Tiled images can be easily cached in a client-side web browser, in the data server, or on a specific cache server. The temporarily saved data can be used to accelerate the response to a user request when certain image tiles are frequently requested or they are requested by multiple users at the same time. With the standardization of tiling services, the implementation of caching services also became standardized. Main implementations include TileCache for WMS-C, GeoWebCache by Geoserver and MapProxy.

3 Methodology

This chapter will focus on the workflows and relevant data used in the conducted study. As mentioned in the first chapter, the research object is to colourize the grey-scale CORONA images taken from the 1960s to 1970s in the region of Saxony and visualize the results. The process can be divided into three parts: colourization, evaluation and visualization. A deep-learning-based method will be used to colourize the CORONA images, and evaluation of the results will be performed both quantitatively and qualitatively. The obtained results together with other relevant data will be displayed in a web mapping application. Detailed workflows are described in the following sections.

3.1 Data description

The images to be colourized are greyscale satellite images produced during the photo-satellite reconnaissance program called CORONA initiated by the U.S. government. These images are georeferenced film strips. Each film strip follows a northwest-southeast direction and covers a wide area. The film strips used in this study are all taken on 10th August 1975, covering the whole region of the Saxony state of Germany. Each film strip is further cut into smaller patches, with overlaps between adjacent patches.

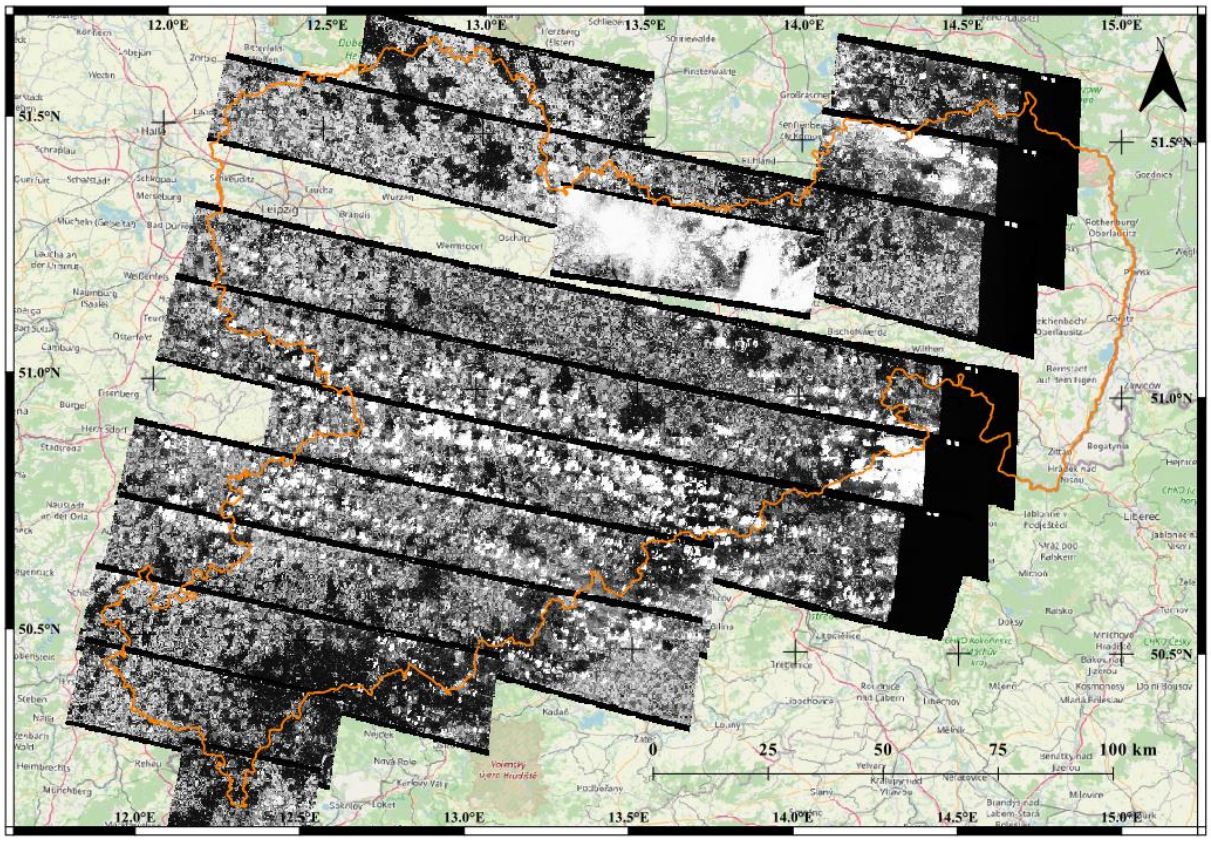


Figure 13: An illustration of overlapping film strips

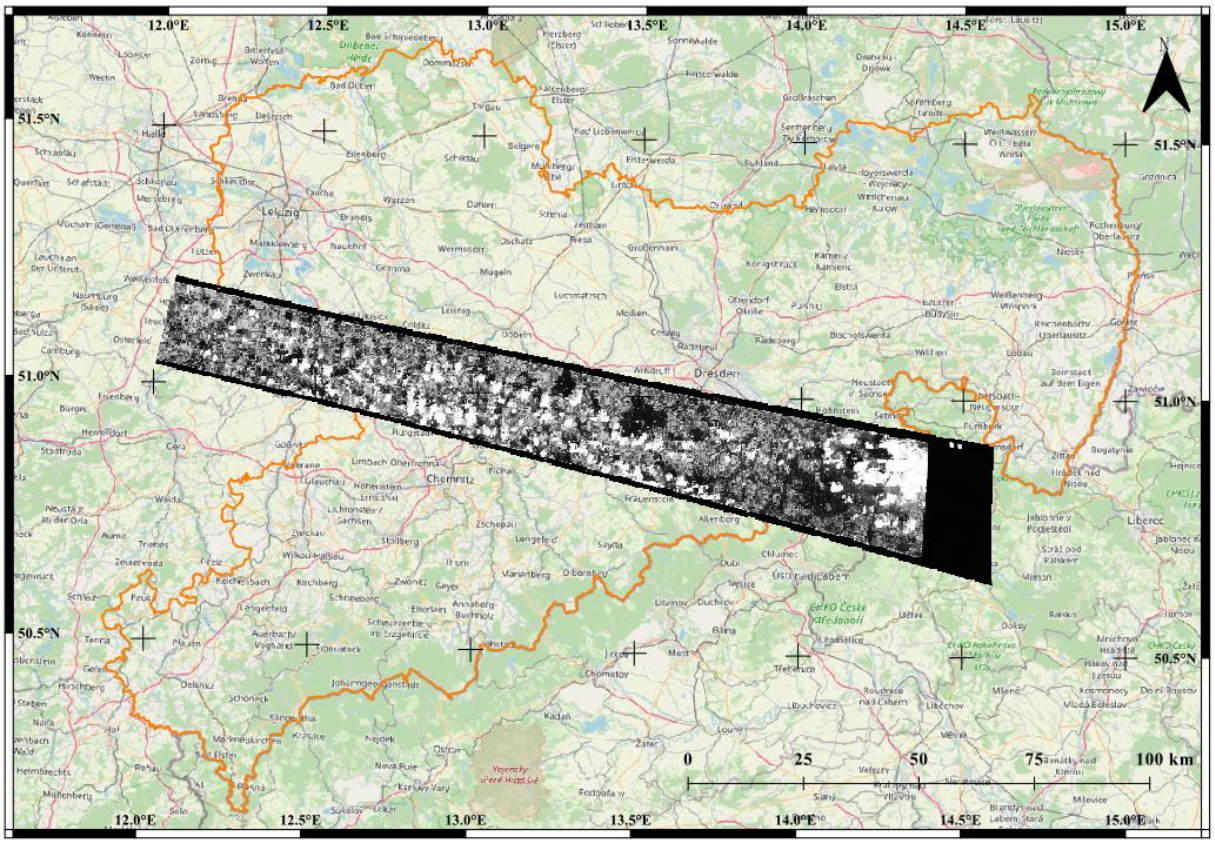


Figure 14: One of the film strips used in the study, covers the middle region of Saxony

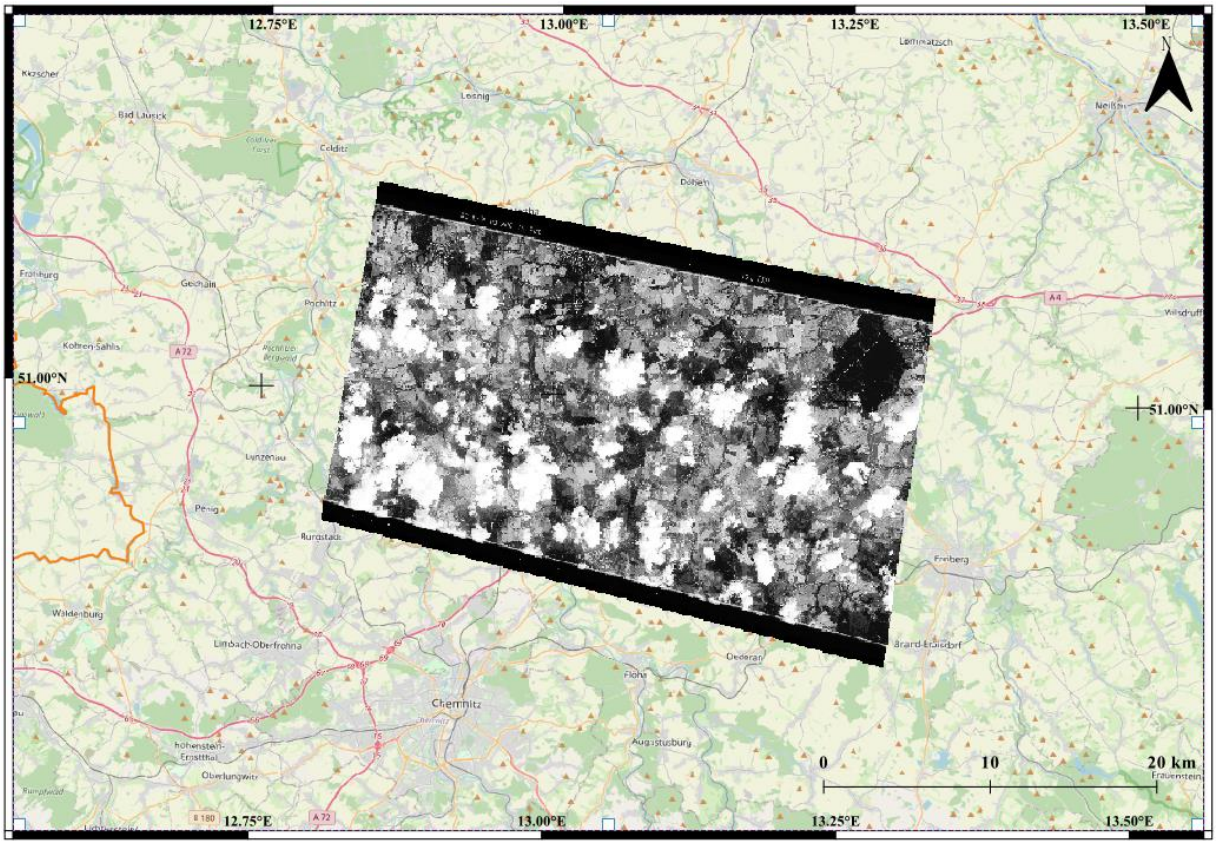


Figure 15: A smaller image patch from one film strip

In total, the CORONA satellite image dataset consists of 41 greyscale images derived from 12 film strips side overlapping between adjacent ones. Each image has a spatial resolution of $2m$. These images will be used to select suitable patches for the training data.

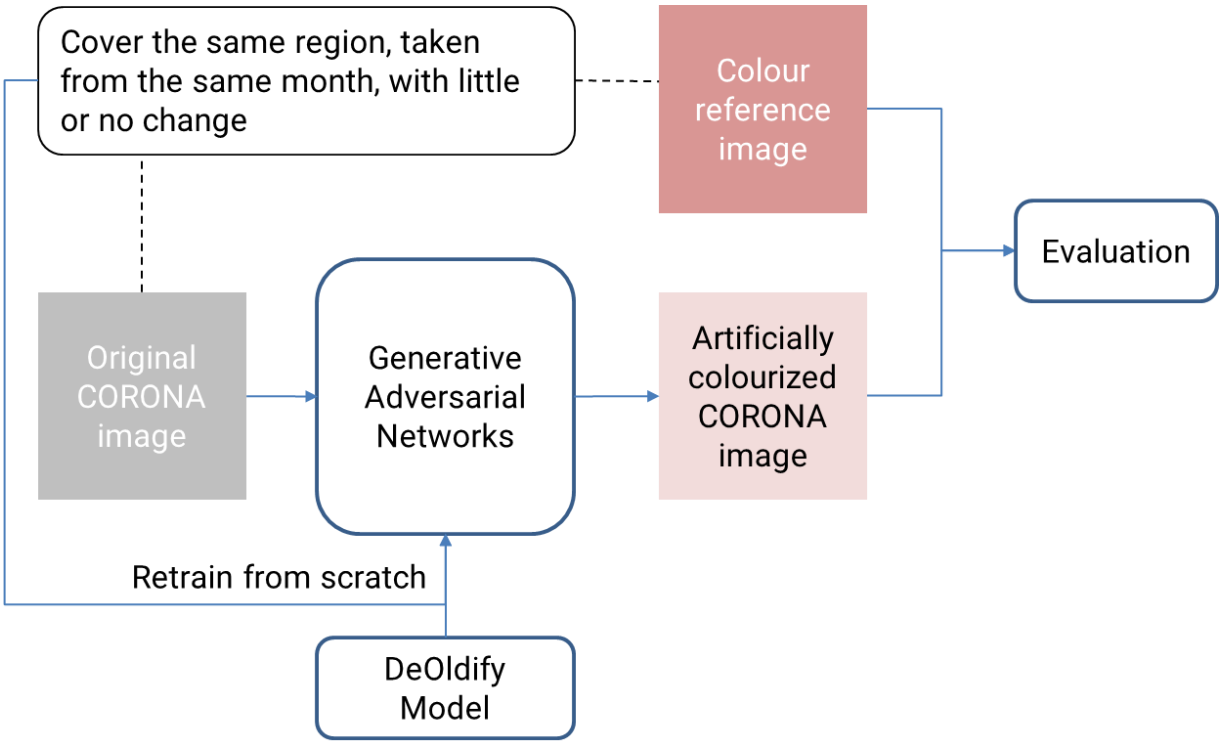


Figure 16: Workflow for image colourization

As shown in Figure 16, the workflow of the image colourization process requires two datasets. Besides the CORONA images to be colourized, a set of colour reference images are also needed. These colour reference images should meet the following requirements:

- cover the same regions as the CORONA image
- taken in the same period or better from the same month as the CORONA image
- have the same spatial resolution as the CORONA image
- the covered area should have little or no changes in terms of image content

The colour reference images are taken from Staatsbetrieb Geobasisinformation und Vermessung Sachsen (GeoSN). GeoSN provides digital orthophotos at a ground resolution of $0.2m$, obtained through UAV flights with a digital camera. Since 2010, these digital orthophotos come in the forms of panchromatic (black and white), three-channel RGB and four-channel RGB-Infrared (GeoSN). In this study, the three-channel RGB digital orthophotos will be used.

3.2 Data pre-processing

As shown in Figure 16, a GAN will be trained for the image colourization process. Thus, these datasets need to be pre-processed into suitable training data. Firstly, the original CORONA satellite images will be cropped to suitable study regions.

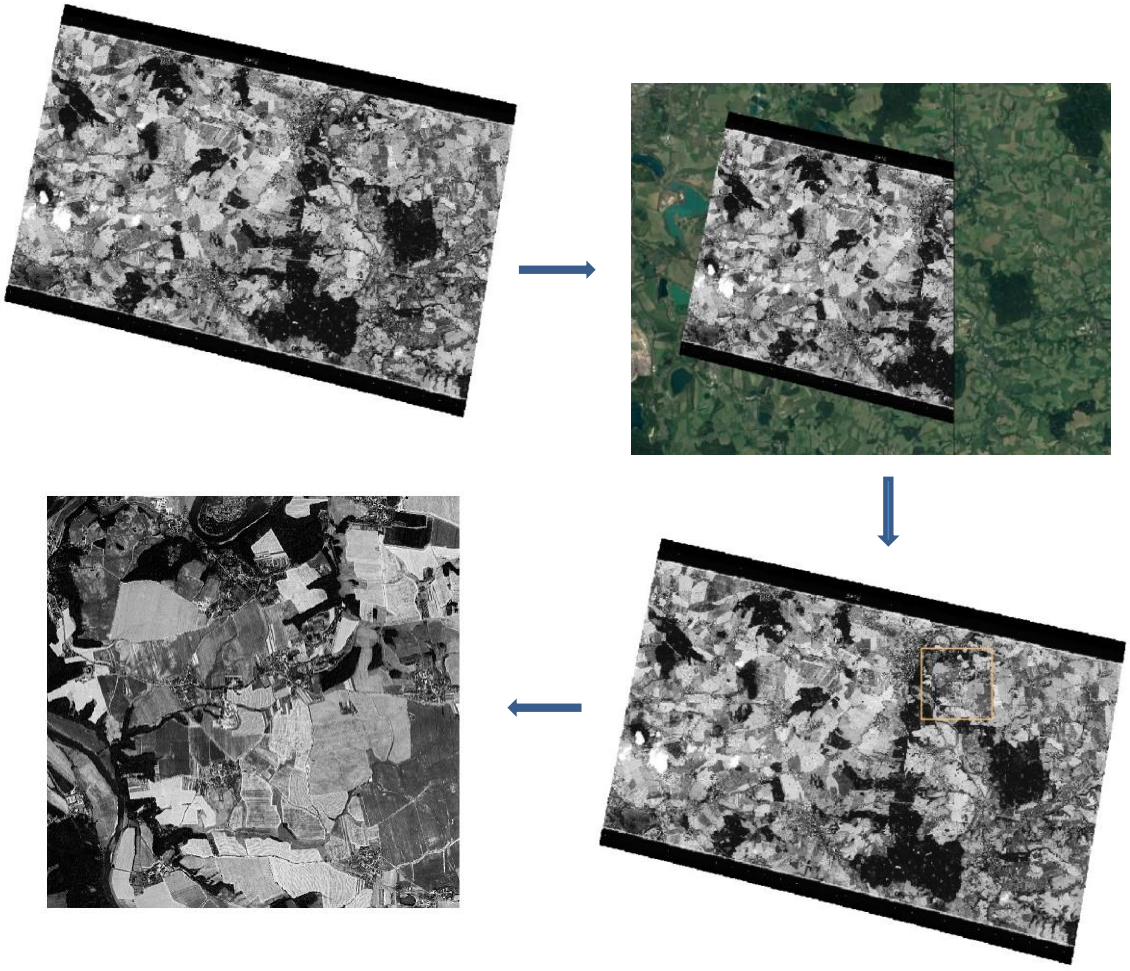
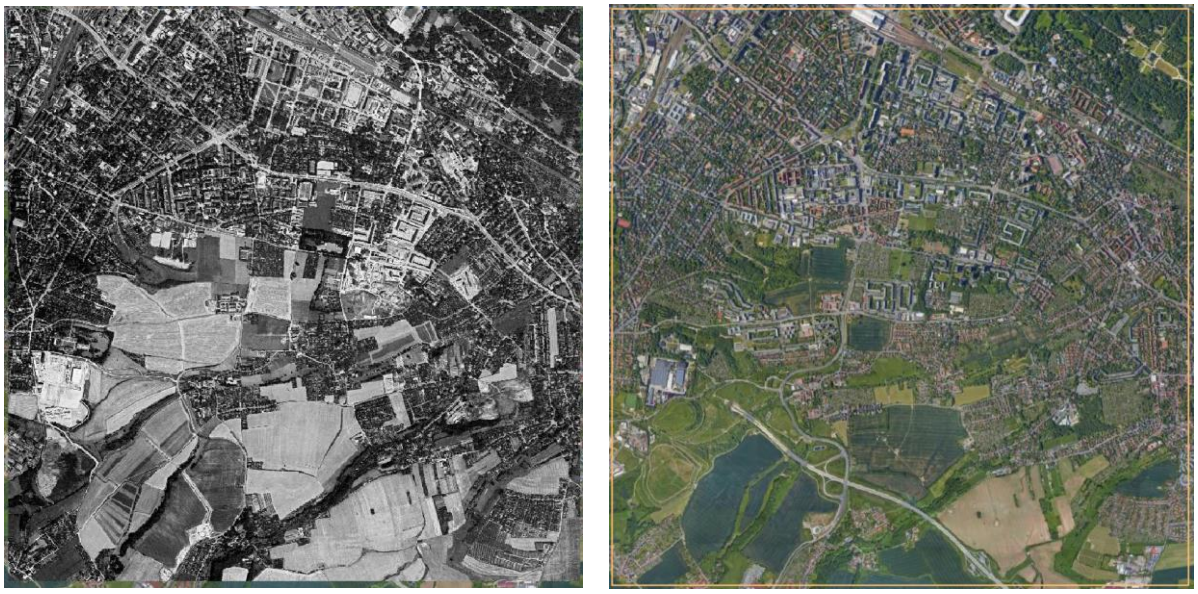


Figure 17: The process of selecting suitable study regions

As shown in Figure 17, a CORONA image will be overlaid with the latest satellite image from Google Maps, a swipe tool is used to switch between the two layers. The idea is to find a suitable region that has little or no change in terms of landscape and land use between 1975 and recent times. This heavily relies on visual comparison and is a time-consuming process. To get a good result in colourization, the selection of training data should contain as many land cover types as possible. Common land cover types in the CORONA images include agricultural land, forest, water bodies and urban land. It is relatively easier to find areas of agricultural land or forest that meet the requirements. But in urban land, many cities have already had huge changes since 1975. The selection

for urban land is significantly harder and thus the training data contains fewer images that cover urban land.

(a)



(b)

Figure 18: Examples of suitable and unsuitable areas for the study region

Figure 18 shows two examples of choosing the suitable areas. On the left are CORONA images from 1975, and on the right are the latest satellite images from Google Maps. Images on the same row cover the same area. The area covered in (a) is suitable for being chosen as the study region because most of the area did not change. The patterns of landscape and land cover are well preserved. While in Figure 17 (b), although the upper part of the area in the image did not have noticeable changes, the bottom

part significantly changed. Buildings have taken the place of agricultural land and new roads were built.

In total, 20 study regions are selected, each of them is an image of 2240 x 2240 pixels, with a ground resolution of 2 meters. The choice of width and height will be explained in the following section. As the workflow in Figure 16 shows, the training data consists of two parts: the original CORONA images and colour reference images. Satellite images from Google Maps do not have a uniform ground resolution, and it is not clear what is the exact ground resolution of an image tile at a certain place. As a result, those images are only used for selecting suitable study regions of CORONA images. Instead, the colour reference images are downloaded from GeoSN product download platform. All digital orthophotos are organized in tiled grids, and by selecting the grids, corresponding image tiles will be downloaded. Then the downloaded image tiles are merged, cut to the extent of the corresponding CORONA image, and resampled to the same ground resolution.

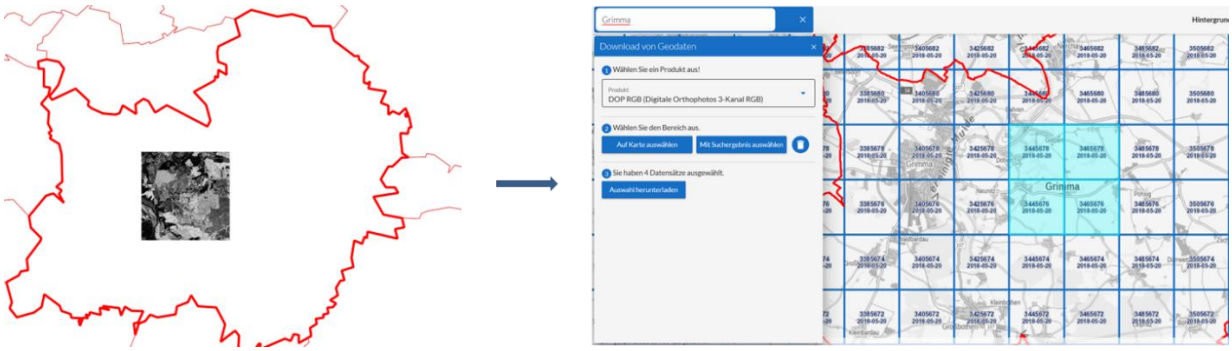


Figure 19: An illustration of the process of downloading digital orthophotos

So far, the images are all at a size of 2240 x 2240 pixels. This is not the desired size for training data of the neural network. Therefore, these images need to be further cut into smaller tiles. This process is done through the Export Training Data For Deep Learning geoprocessing tool in ArcGIS Pro. This tool can convert labelled vector or raster data into deep learning training data. In this study, the training data is greyscale CORONA images, and the label is corresponding colour digital orthophotos. This tool will cut each image pair into smaller image tile pairs of fixed tile size and tile stride. As shown in Figure 20, the input raster is the grayscale CORONA image, and the colour digital orthophoto will be added as an additional input raster. Tile sizes X and Y decide the tile size, and stride X and Y decide the overlap between each tile. Here all four parameters are set to 224. Thus, each study region will be cut into 100 image tiles with a size of 224 x 224 pixels and no overlapping with each other. In total, 2000 labelled images will be generated for deep learning.

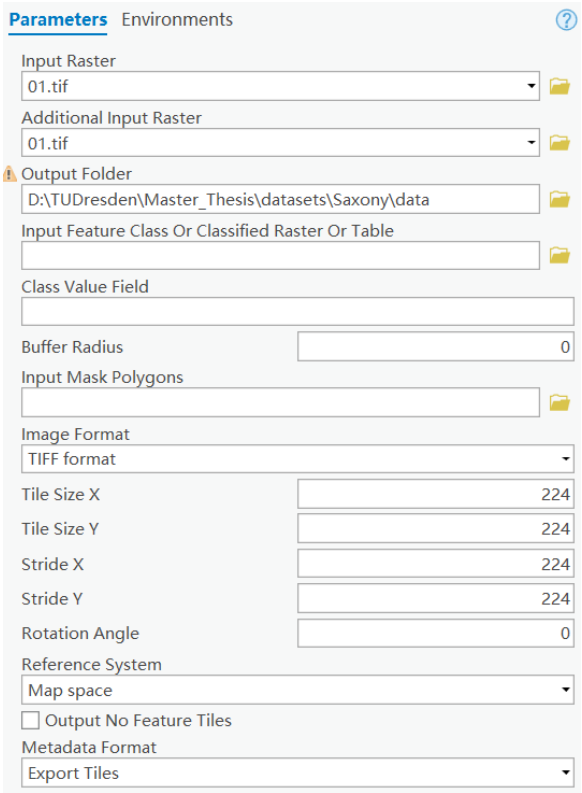


Figure 20: Export Training Data for Deep Learning tool

Then these labelled images will be organized into an ImageNet-like dataset. Two folders contain greyscale images and colour reference images respectively. In each folder are 20 subfolders representing each study region, and inside each subfolder are the image tiles generated from the same region. The greyscale image and its label (colour reference image) share the same file name and are placed in the subfolders with the same fold names.

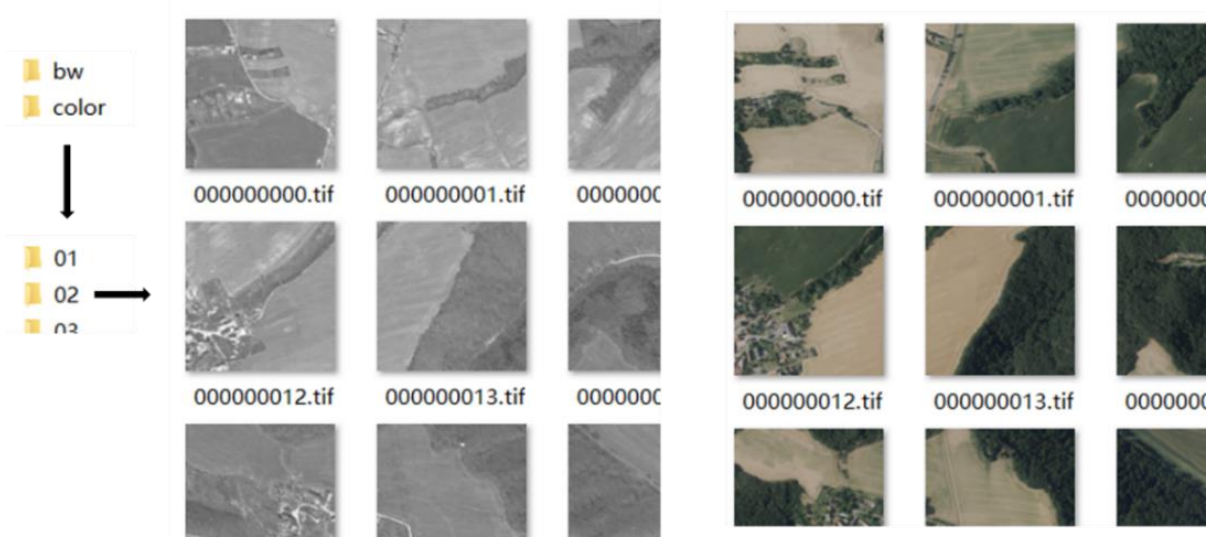


Figure 21: Organization of data

3.3 GAN and the DeOldify model

In this study, a deep learning model based on Generative Adversarial Network (GAN) will be used to colourize the greyscale CORONA images. GAN was first introduced by Goodfellow et al. (2014). The framework is that two models are simultaneously trained: one generative model G and one discriminative model D (also referred to as a critic C). The generator G tries to generate new data according to the input. Critic C tries to discriminate fake data generated by G from real data. The framework tries to solve the following two equations:

$$\log\big(D(y)\big) + \log\left(1 - D(G(x))\right) \tag{2}$$

$$\log\left(1 - D(G(x))\right) \tag{3}$$

Where x is the input data, y is the real data, $G(x)$ is the fake data generated by G, and $D(y)$ is the probability that the critic C labels the given data as real. The critic tries to maximize Eqn. (2) while the generator tries to minimize Eqn. (3).

In image colourization problem, the generator G takes greyscale images as input and generates colour images. The critic's job is to correctly label all colour images generated by G as fake, and colour images from real-world data as real. At the early stage of training when the generator performs poorly, the critic can easily distinguish between real and fake images. But as the training continues, the generator can generate images of higher quality, thus making it more difficult for the critic to correctly discriminate. As the critic gains better performance through more training, the generator is harder to produce images that could fool the critic. Ideally, the competition between these two models would eventually lead to equilibrium and produce plausible colour images that almost look like real ones.

This study will exploit the DeOldify model to colourize the greyscale CORONA satellite images. The DeOldify model was proposed by Jason Antic (Antic, 2018) and is dedicated to restoring and colourizing old black and white photos. Originally the DeOldify model was trained on the ImageNet dataset (Deng et al., 2009), which does not contain many satellite images. To achieve a better result in satellite image colourization, the DeOldify model will be retrained on the previously mentioned CORONA image dataset. The model is based on GAN architecture. Details about the model will be given in the following sections.

3.3.1 Self-attention

Self-attention was first introduced in the Self-Attention Generative Adversarial Network (SAGAN) proposed by Zhang et al. (2019). In traditional convolution networks, high-level features are only generated using low-level local features due to the limited size of the local receptive field of a convolution kernel. Self-attention intends to reconstruct image details not only from neighbouring pixels but also from long-range points in distant regions in the image.

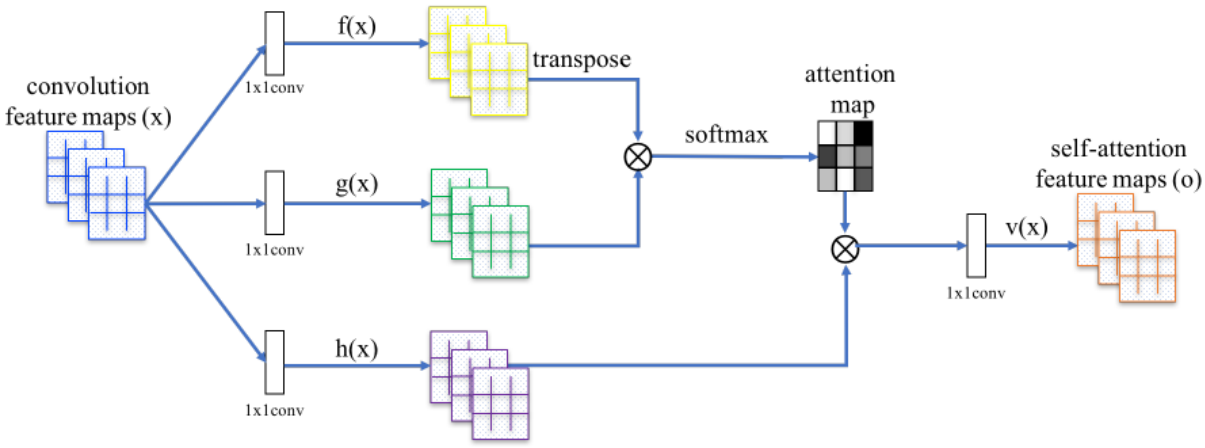


Figure 22: An illustration of self-attention (Zhang et al., 2019)

As shown in Figure 22, the input feature maps x are firstly transformed into two feature spaces f and g. This process is done by a 1x1 convolution kernel and the outputs are denoted as $f(x)$ and $g(x)$ respectively. Then the two output feature maps are multiplied and generate an attention map. This attention map can be expressed as the following equation:

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})}, where\ s_{ij} = f(x_i)^T g(x_j) \tag{4}$$

The element value $\beta_{j,i}$ in the attention map denotes the extent to which the model attends to the pixel i when synthesizing the region j to construct the features. The attention map is further multiplied by an output $h(x)$ from a 1x1 convolution and then passed through a final 1x1 convolution to generate the self-attention feature maps. The outputs can be expressed in the following equation:

$$o_j = v\left(\sum_{i=1}^{N} \beta_{j,i} h(x_i)\right), h(x_i) = W_h x_i, v(x_i) = W_v x_i \tag{5}$$

In addition, the generated feature maps are multiplied by a scale factor and then added back to the input. The final output can be expressed as:

$$y_i = \gamma o_i + x_i \tag{6}$$

Initially, γ would be set to 0 and modified through the training process. This encourages the model to first rely on features in local regions and gradually attend to distant regions.

Figure 23: Illustrations of how features are generated with self-attention (Zhang et al., 2019)

Figure 23 shows input images with pixels encoded in different colours. And the following images in each row denote which regions each pixel attends to when reconstructing image features. Similar patterns in distant regions can be correctly detected and used for image feature generation with self-attention. In the DeOldify model, self-attention is both implemented in the generator and critic. In generator, this helps the network to generate fake images with rich and consistent detailed information. And critic can also benefit from self-attention as it allows the critic to identify features more easily and accurately thus improving the performance in telling apart real images and fake images.

3.3.2 Generator

3.3.2.1 Skip connection and U-Net

Skip connection was first introduced in Deep Residual Network (ResNet) proposed by He et al. (2016). Deep convolutional neural networks have been proven to have better performance on image classification and other visual recognition problems. But simply stacking more layers in the neural network will not lead to a better result. In fact, as the depth of the network increases, accuracy gets saturated and then quickly degrades.

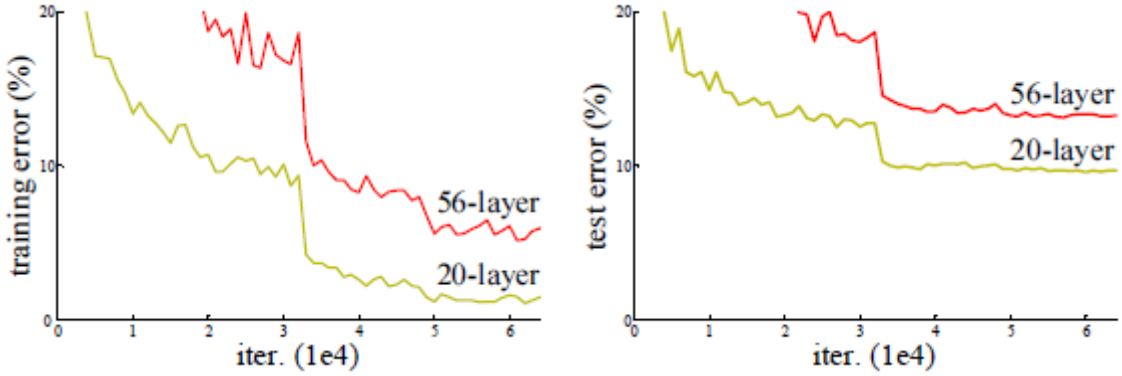


Figure 24: Training and test error of two CNNs with different depths (He et al., 2016)

To solve the degradation problem occurring in deep networks, a deep residual learning framework was introduced. Instead of simply stacking more layers, a "shortcut connection" is used between newly stacked layers. This "shortcut connection" can be represented as a residual network block, it is defined as:

$$y = F(x, \{W_i\}) + x \tag{7}$$

$$F(x, \{W_i\}) = W_2\sigma(W_1 x) \tag{8}$$

Where x is the input of the residual network block, y is the output of the residual network block, F is the mapping function to be learned, σ is the Rectified Linear Unit (ReLU) activation function, and W_i is the weight matrix at the i^{th} layer. In convolutional networks, the weight layer is a filter with a fixed kernel size. This is based on the assumption that the spatial size of the input data x and $F(x)$ is the same. If not, the addition will be performed with a stride of 2 to match the size of the output $F(x)$, and extra paddings of 0 will be used for increased channels.

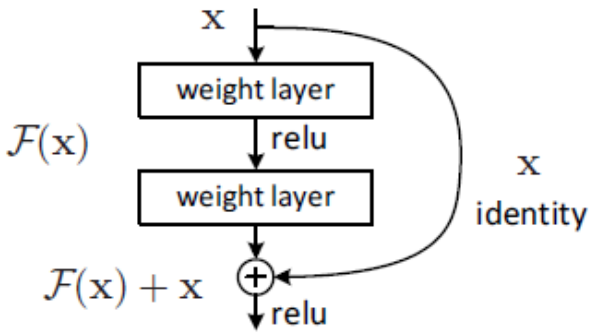


Figure 25: A typical residual network block (ResNet block) (He et al., 2016)

As shown in Figure 25, the input data x is passed through two weighted layers (each weighted layer typically consists of a 3x3 convolution layer and a batch normalization

layer), generating an output $F(x)$, and then a simple elementwise addition is performed on the output with the original input x. The final output of this block is $F(x) + x$. This simple addition is also referred to as identity mapping. This block can efficiently solve the degradation problem because, with skip connection, the intermediate weight layers can always be learned to set to zero if the final output performs worse than without these additional layers. That is to say, the output of the block $F(x) + x$ will be x when the weight in each layer is all 0 because then $F(x)$ will also be 0. And these intermediate layers will have no negative effects on the network. Therefore, the newly added layers can deepen the neural network without leading to worse performance.

Skip connection can also be found in U-Net (Ronneberger et al., 2015). U-Net was originally proposed to solve the problems of massive training data requirements for deep neural networks and labels lacking localization. In conventional CNNs, the labels of training image data are often single-class labels. But in fields like biomedical image processing, the label for a training image should be another image, meaning the final output of a neural network should also be an image instead of the probability of an image belonging to a certain class.

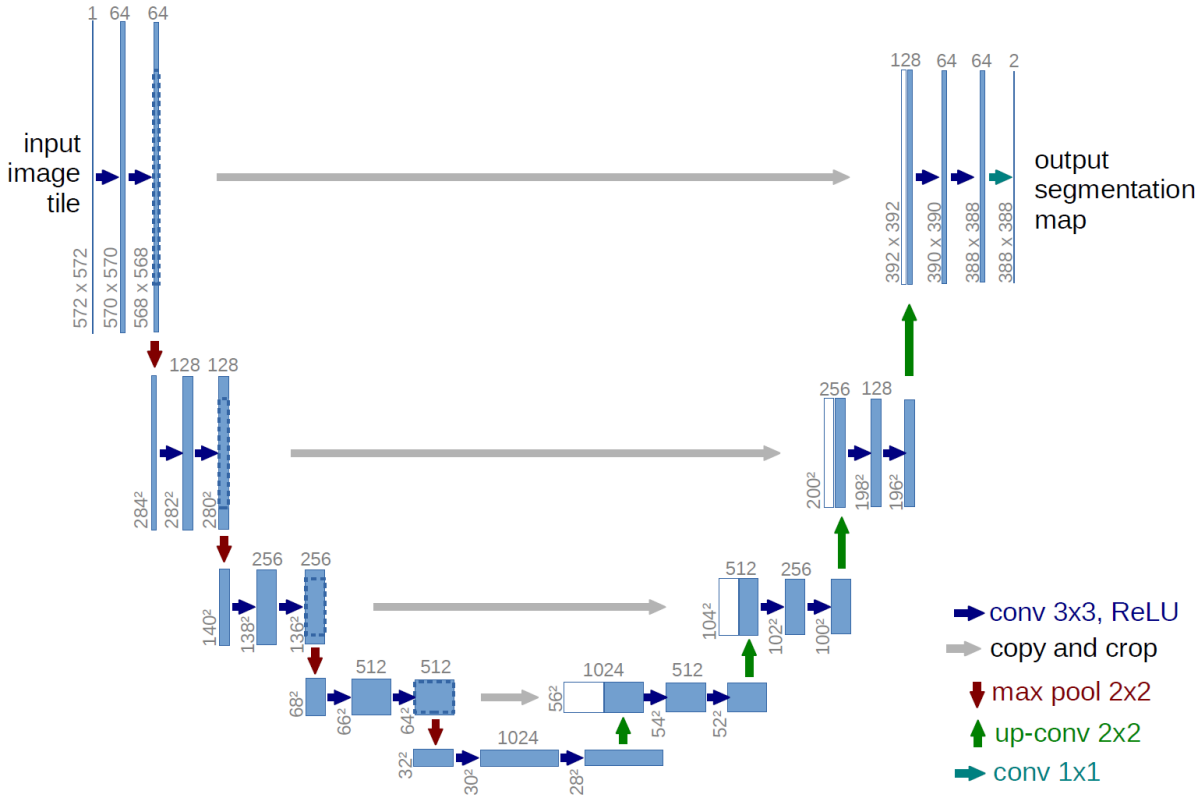


Figure 26: U-Net architecture (Ronneberger et al., 2015)

The proposed U-Net architecture by Ronneberger et al. (2015) is shown in Figure 26. This convolutional neural network mainly takes two paths: a downsampling path and an upsampling path. Each blue block represents a multi-channel feature map after convolution with a filter of kernel size 3. In the downsampling path, a 2 x 2 max pooling layer is used to reduce the spatial size of the feature maps, while in the upsampling path, an up convolution operation is used to increase the spatial size of the feature maps. Transposed convolution (Dumoulin & Visin, 2016) is a commonly used operation to achieve this purpose. Skip connection happens in the downsampling path before the max pooling layer, i.e., before the spatial size of the feature map dramatically halves. Unlike in ResNet using simple addition for skip connection, the feature map is concatenated to the new feature map. Skip connection keeps feature maps extracted in the early stage of the convolution, thus containing rich low-level and mid-level feature information, and these feature maps are used for reconstructing the image details in the upsampling path. It has been displayed in Isola et al. (2017) that skip connection in U-Net largely improves the output performance.

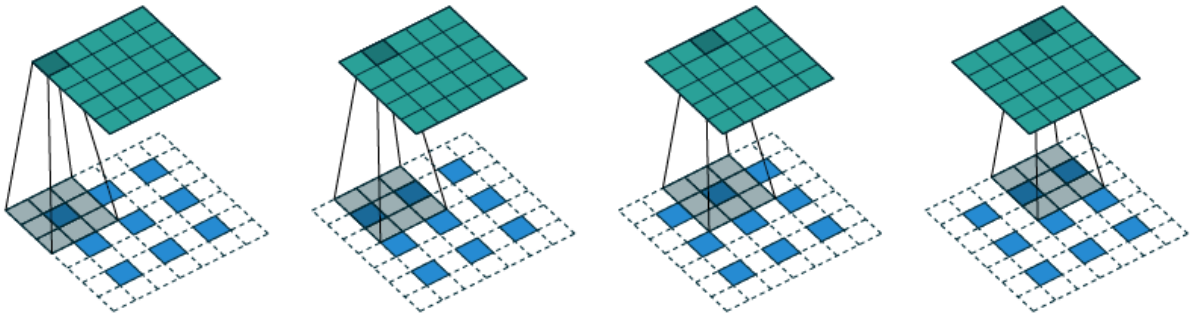


Figure 27: An illustration of how transposed convolution works (Dumoulin & Visin, 2016)

However, transposed convolution is known to bring the so-called checkboard artefacts (Sugawara et al., 2018). A sub-pixel convolution method called periodic shuffling was proposed by Shi et al. (2016). This method was first introduced to solve image super-resolution problems. It upscales an image of size H x W x C to size rH x rW x C where r is the upscale factor, H and W are height and width respectively, and C is the number of channels. To achieve this upscaling, the input image or feature maps of size H x W x C will be passed through L-1 convolution layers which do not change the spatial size (height and width), but only the number of channels. The last layer (L^{th} layer) performs the so-called periodic shuffling operation which can be expressed as the following equations:

$$I^{SR} = f^{L}(I^{LR}) = \mathcal{PS}(W_L * f^{L-1}(I^{LR}) + b_L) \quad (9)$$

$$\mathcal{PS}(T)_{x,y,c} = T_{\lfloor x/r \rfloor, \lfloor y/r \rfloor, C \cdot r \cdot mod(y,r) + C \cdot mod(x,r) + c} \quad (10)$$

The last layer has a convolution kernel W_L which outputs feature maps of $C \cdot r^2$ channels, the PS denotes the periodic shuffling operation that rearranges the pixel positions, thus upscaling feature maps of size H x W x $C \cdot r^2$ to size rH x rW x C as shown in Figure 28.

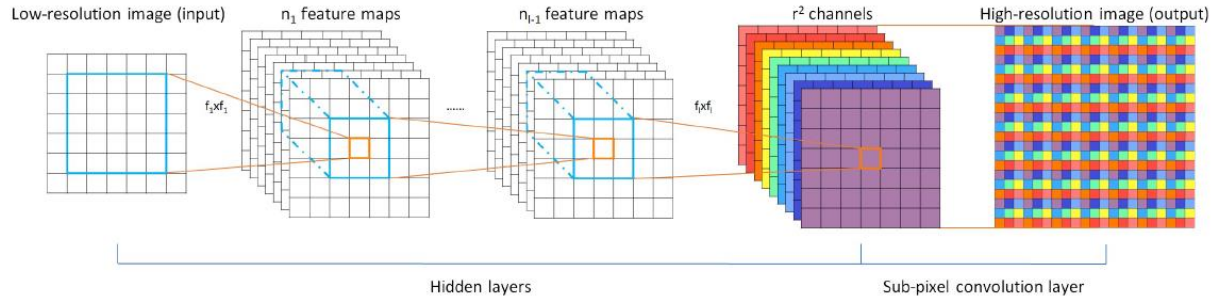


Figure 28: An illustration of sub-pixel convolution

In this study, the generator of the DeOldify model is a pre-trained U-Net. Different from the original U-Net model, the left part, i.e., the downsampling path, is a ResNet model pre-trained on the ImageNet dataset (Deng et al., 2009), which consists of 3-channel colour images of size 224x224 pixels. The DeOldify model provides two modes for image colourization: artistic and stable. Artistic mode tends to generate images with more vivid colours while stable mode has more stability in terms of image quality. The main difference between the two modes is that they use a ResNet cut to the last layer before the fully-connected layer with a depth of 34 layers (artistic) and 101 layers (stable). In this study, the artistic mode will be used due to GPU memory limits.

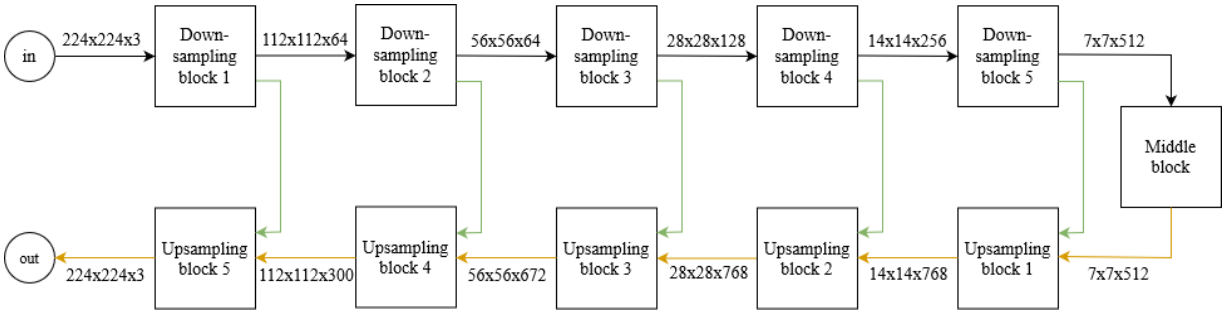


Figure 29: The architecture of the generator used in this study

The architecture of the generator for artistic mode is shown in Figure 29. The downsampling path is identical to that of the ResNet 34 model without the last fully-connected layer. Each downsampling block consists of several repeating ResNet blocks as shown in Figure 25. The upsampling path is made of several upsampling

blocks in which periodic shuffling operation is implemented for upscaling. Self-attention layer is also inserted in upsampling blocks to enhance the feature reconstruction. Skip connection happens after upscaling in each upsampling block. The feature maps from the downsampling path will be concatenated to the output after upscaling.

3.3.3 Critic

Compare with the generator, the critic has a simpler architecture. The critic is a classic binary classifier. The main job of the critic is to distinguish between real image data fetched from the dataset and fake image data generated by the generator.

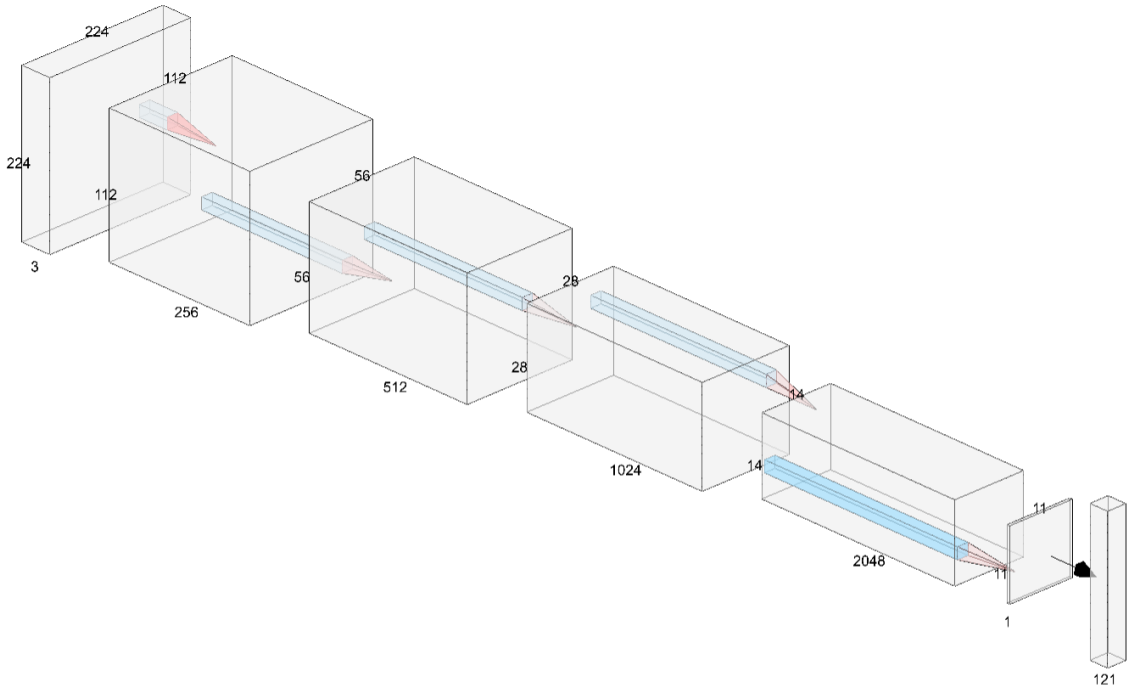
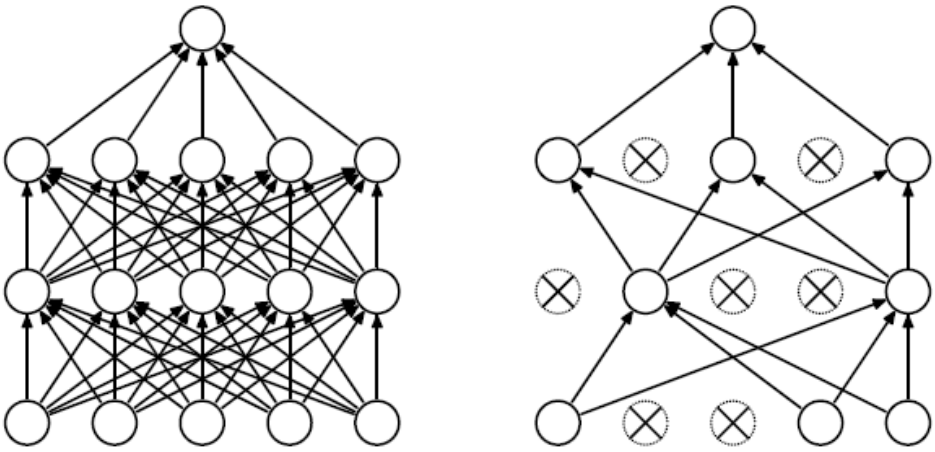


Figure 30: The architecture of the critic

As shown in Figure 30, the critic follows a classic CNN style. It consists of several layer blocks composed of a convolution layer followed by an activation layer. Self-attention layer is also inserted into certain layer blocks to enable the critic to identify features more easily. A dropout layer (Srivastava et al., 2014) is inserted in between layer blocks. It was introduced to prevent the neural network from overfitting by randomly dropping out nodes or one entire channel in feature maps from the input to the next layer. Besides, self-attention is also implemented in the critic. All downscale operation is achieved by using a convolution of stride 2 instead of a pooling layer.



(a) Standard Neural Net (b) After applying dropout.

Figure 31: An illustration of how dropout works (Srivastava et al., 2014)

3.3.4 Loss function

3.3.4.1 Critic loss

The critic loss is the binary cross entropy with logit loss. The loss function can be expressed as the following equation:

$$Loss_C = -\frac{1}{N}\sum_{n=1}^{N} w_n[y_n \cdot \log\big(\sigma(x_n)\big) + (1 - y_n) \cdot \log(1 - \sigma(x_n))] \tag{11}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{12}$$

Where N is the batch size. x_n is the output of the critic network. It should be a value between 0 and 1, which represents the probability of the input image is real. y_n is the corresponding label (fake or real) of the input image. It is either 0 or 1, with 1 for a real colour image and 0 for an artificial colour image. w_n is the weight for each image in a training batch. In this study, it is set to 1 as all images in a batch contribute the same. σ denotes the sigmoid function. It transforms the value from $(-\infty, +\infty)$ to $(0, 1)$.

3.3.4.2 Generator loss

The loss function of the generator network is perceptual loss based on a pre-trained VGG-16 model. This method was first proposed by Johnson et al. (2016). It proves to work better than per-pixel loss such as Root Mean Square Error (RMSE) and Peak Signal-to-Noise Ratio (PSNR) in image transformation problems such as image super-resolution, style transfer, and image colourization.

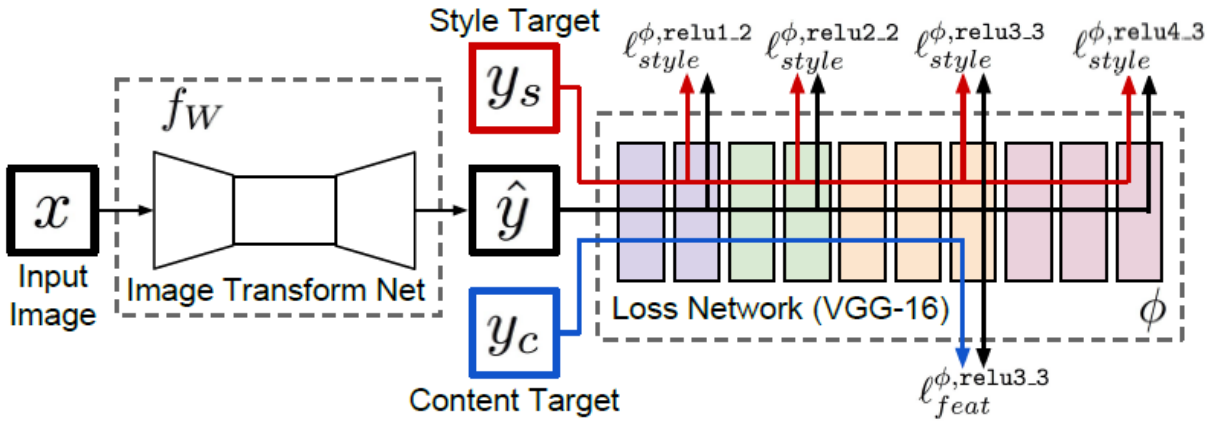


Figure 32: Overview of how to calculate perceptual loss (Johnson et al., 2016)

As shown in Figure 32, the perceptual loss is calculated using a loss network based on pre-trained VGG-16 architecture. The output $\hat{y}$ from an image transform net (for example, image colourization network) is passed to the loss network together with the style target y_s and content target y_c. In image colourization network, only the content target is concerned so the style target can be omitted in this case. The output $\hat{y}$ would be an artificially colourized image and the target content y_c would be the corresponding real colour image of the input greyscale image. Both images are passed through a set of convolution layer blocks. The output feature maps of a certain activation layer in the loss network are extracted and used for the calculation of perceptual loss. The advantage of using perceptual loss rather than the per-pixel loss is that per-pixel loss is heavily influenced by pixel position shift. Two identical images with only a one-pixel position shift in height would result in a large per-pixel loss despite they have the same image content. But using perceptual loss would avoid this situation as perceptual loss compares the feature information of the images and is location irrelevant.

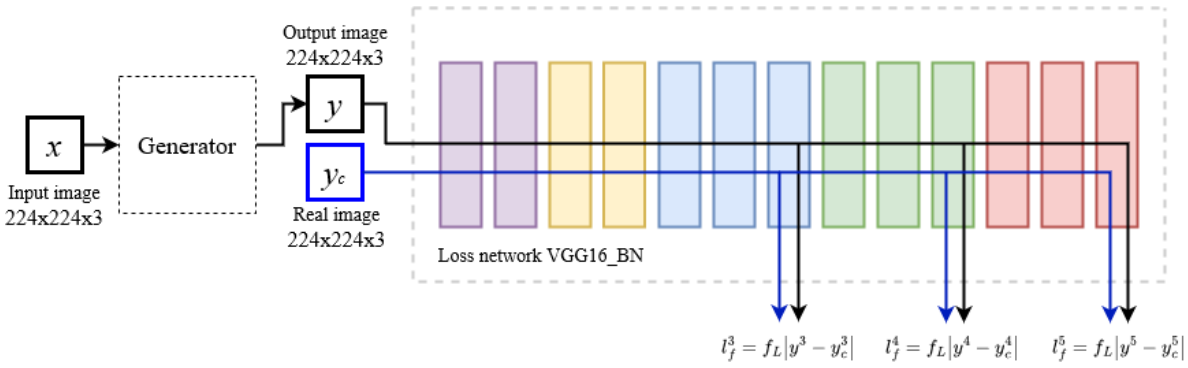


Figure 33: Feature loss of generator

As shown in Figure 33, in this study, the loss network is a pre-trained VGG-16 network with batch normalization layers. Each coloured block denotes a series of a 3x3 convolution layer, a batch normalization layer and an activation layer. Different colours denote the spatial size is halved by a max pooling layer. Feature maps outputted by the last layer of the three chosen blocks are used to calculate the corresponding L_1 loss value. The final perceptual loss is the sum of the weighted average of the three loss values and the L_1 loss value of y and y_c. This can be given as:

$$Loss_{G\_net} = f_L(y, y_c) + \sum_{i=3}^{5} \gamma_i f_L(y^i, y_c^i) \tag{13}$$

$$f_L(u, v) = \frac{1}{MN} \sum_{j=1}^{M} \sum_{k=1}^{N} |u_{j,k} - v_{j,k}| \tag{14}$$

While in GAN training, the generator loss consists of two parts: the generator network loss and the critic fake predict loss. The generator takes one more loss of the probability of the critic classifying a fake colour image to be real. The total loss for the generator can be given as follows:

$$Loss_G = \alpha \cdot Loss_{G\_net} + \beta \cdot \frac{1}{N} \sum_{n=1}^{N} -w_n [y_n \cdot \log(\sigma(x_n))] \tag{15}$$

Where α and β are weights for the two parts. N is the batch size. x_n is the output of the critic network for a fake colour image generated by the generator. A higher value means that the critic is more likely to label this fake colour image as real. y_n is the corresponding label of the input image. In this case, it is set to 1. w_n is the weight for each image in a training batch. In this study, it is set to 1 as all images in a batch contribute the same. σ denotes the sigmoid function. In this case, when the probability of a fake colour image being real increases, the calculated loss decreases and thus leading to a smaller total loss.

3.3.5 Training settings

The training of a GAN takes an alternative approach. Training is switched between the generator and critic repeatedly. The steps are:

1. Freeze the generator and train the critic for one or more steps:
   - Get one batch of real images from the dataset
   - Get one batch of fake images generated from the generator

- Send the two batches of images to the critic for training
- Update the parameters of the critic according to the critic loss in Eqn.11 and gradients

2. Freeze the critic and train the generator for one step:
   - Generate one batch of fake images from the input images
   - Pass the fake images to the critic to evaluate
   - Update the parameters of the generator according to the total generator loss in Eqn.15 and gradients
3. Repeat steps 1 and 2 alternatively until the network converges.

When the network is set to freeze, it is put in evaluation mode and the parameters will not be updated.

The DeOldify model mainly implements two techniques to optimize the training process: the Two Time-scale Update Rule (TTUR) and NoGAN training.

The Two Time-scale Update Rule (TTUR) was proposed by Heusel et al. (2017) to address the convergence problem of GAN training. GAN training is unstable and may not always lead to convergence. However, it has been proved in Heusel's study that by setting different learning rates for generator and critic respectively, GAN will converge to a local Nash equilibrium. It can be expressed in the following equations:

$$\omega_{n+1} = \omega_n - b(n) \cdot \tilde{g}(\theta_n, \omega_n) \tag{16}$$

$$\theta_{n+1} = \theta_n - a(n) \cdot \tilde{h}(\theta_n, \omega_n) \tag{17}$$

Where ω_n and θ_n denote the parameters of critic and generator respectively. $\tilde{g}$ and $\tilde{h}$ denote the stochastic gradients used to update parameters for the critic and generator. $b(n)$ and $a(n)$ are learning rates for critic and generator. These parameters are updated at each iteration. It is also suggested that the critic should have a higher learning rate than the generator and takes more steps at one time. It is also suggested that the critic should have a higher learning rate than the generator and more steps at one time when training alternatively.

In the actual training setting of the DeOldify model, the learning rate of the critic is multiplied by a factor of 5 by the learning rate of the generator. The switch between training critic and generator is decided by a critic loss threshold T. The training starts from the critic and will be switched to the generator only if the critic loss is below threshold T. Then the model will train the generator for one step and switch back to the critic.

NoGAN training is a new training strategy proposed by Antic et al. (2019) to accelerate the training of GAN and pursue a more stable output. Instead of directly training the entire GAN, the generator and critic will be pretrained separately and then trained in a GAN together. The steps are described as follows:

1. Pretrain the generator. When pretraining the generator, only the perceptual loss in Eqn.13 is used for updating the parameters. The generator is trained in a conventional way of training a convolutional neural network, thus the convergence of the network is easier than in a GAN training setting. This step aims to train the generator to produce images of high quality in terms of feature reconstruction rather than colours. It is supposed to be the most time-consuming part of the whole NoGAN training process.
2. Save the images generated by the pretrained generator.
3. Pretrain the critic. As the pretraining of the generator, the critic is trained as a binary classification network, and it is easier to achieve convergence. This step is to train the critic to achieve high accuracy for correctly labelling real and fake images.
4. Train the generator and critic in a GAN training setting. Train the two networks in an alternative approach as the GAN training. It has been observed that it takes much less time for this direct GAN training before the network produces plausible colour images compared with no pretraining of the two networks separately.

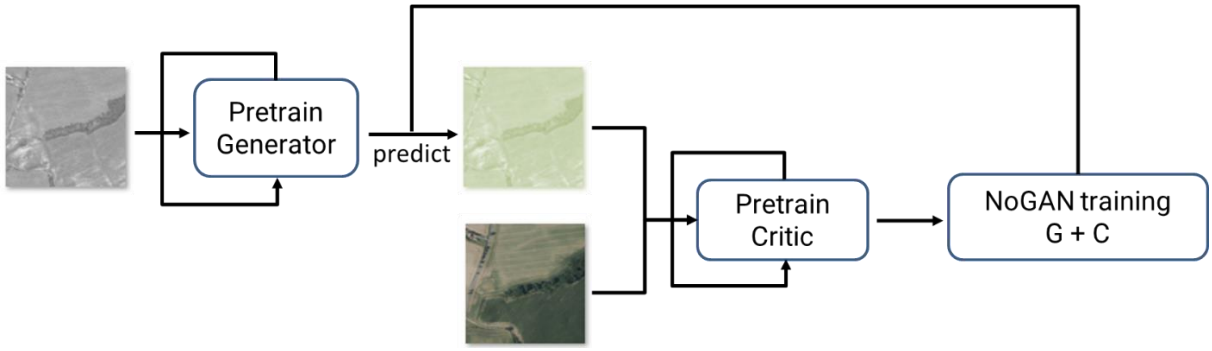


Figure 34: NoGAN training process

In summary, the DeOldify model pretrains the generator and critic separately before direct GAN training and sets different learning rates for the two networks when performing GAN training in an alternative approach.

3.4 Performance evaluation

In neural network training, training loss and validation loss are two commonly used metrics for evaluation. However, they are not intuitive and can be difficult to interpret. In GAN training, the generator and critic are competing against each other. The training loss of one network increases as the loss of the other one decreases. But they cannot directly reflect the performance of the network. In this study, the evaluation of the network output will be conducted directly on the generated images in two aspects: quantitative and qualitative.

3.4.1 Quantitative evaluation

For image processing problems in neural networks, many metrics can be chosen for the evaluation of the final outputs. In this study, Root Mean Square Error (RMSE) and Peak Signal-to-Noise Ratio (PSNR) are used for quantitative evaluation.

RMSE assesses the similarities between two images pixel-wise. For two images I and J of the same size H x W, RMSE can be calculated using the following equation:

$$RMSE = \sqrt{\frac{1}{HW}\sum_{u=1}^{H}\sum_{v=1}^{W}(I_{u,v} - J_{u,v})^2} \tag{18}$$

RMSE is simple and intuitive to compare the numeric similarities. Together with its unrooted variety Mean Square Error(MSE), it is commonly used in image-to-image transformation problems (Guo et al., 2022; Larsson et al., 2016; Shi et al., 2016; Sugawara et al., 2018).

PSNR evaluates the reconstruction quality between two images. For a monochrome image I and its lower quality counterpart J, PSNR can be calculated as follows:

$$PSNR = 20 \times \log_{10}\left(\frac{MAX_I}{RMSE}\right) \tag{19}$$

Where MAX_I is the maximal pixel value in image I. PSRN is also commonly used in image-to-image transformation problems, especially in image super-resolution (Cheng et al., 2015; Larsson et al., 2016; Poterek et al., 2020; Wu et al., 2021).

Due to the lack of ground truth images for the corresponding training CORONA images, the calculation cannot be directly performed on the generated colour images. Instead, the generated colour images will first be transformed from RGB colour space to YUV colour space, where Y channel is the luminance channel. Then the calculation will be

done between the Y channel of the generated colour image and the original greyscale CORONA image. The assumption is that the luminance channel should remain the same when the greyscale image is colourized.

3.4.2 Qualitative evaluation

In image colourization problems, numeric values cannot always truly indicate the generated image quality, as one particular greyscale image can have multiple colourized counterparts besides the ground truth image. The main task for image colourization is to produce plausible colour images rather than recreate the exact original colour images. Thus, it is up to the human eyes to decide if a colour image looks natural.

In this study, a user study is conducted to qualitatively evaluate the generated colour images. The implementation of a user study can also be found in other studies (Cao et al., 2017; Chia et al., 2011; Gupta et al., 2012; Iizuka et al., 2016; Salimans et al., 2016; Zhang et al., 2016). The participants of the user study will be shown a set of colour images composed of real colour images and their artificially colourized counterparts. The participants will be asked the question “Does this image look natural?”. In this way, the plausibility of an artificial colour image can be assessed by human eyes.

3.5 Web visualization

For the third part of this study, a web mapping application will be developed to visualize the obtained colourized CORONA images and provide necessary GIS functionalities including data visualization, management and manipulation. Ideally, the application should consist of two parts: a backend data server that can provide image data storage and optimize data access, and a frontend web map client that can display image data and provide GIS functionalities.

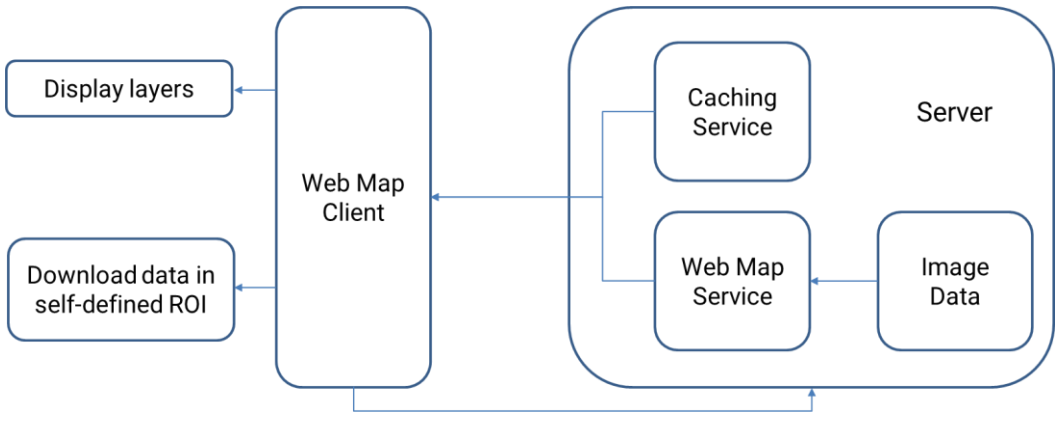


Figure 35: Workflow for the web mapping application

Figure 35 shows the features and workflow of the web mapping application. The server would contain the following features:

- Storing image data (in the form of a raster dataset)
- Visualizing and publishing the image data as a raster layer or a static map image
- Optimizing data access for quicker response

And the web map client would contain the following features:

- Displaying images from different sources as overlays on the web map
- Having a layer control function
- Providing the user with the possibility to download data in a user-defined region of interest (ROI)

The implementation of some key features of this application is described below:

Data visualization and publishing

Stored data in the server is visualized through WMS. A WMS request returns georeferenced static images. When the client side sends a WMS request, certain parameters are required to specify the server address, layer name, extent, size, projection, and other additional information. Vector and raster data are published through WFS and WCS respectively. Vector layers can be retrieved as GeoJSON and other web-compatible formats through WFS from the client side. Similarly, the web client sends a WFS request with parameters specifying data specifications and displaying style (colour, stroke, opacity, etc).

Data caching and tile service

For a quicker response to users' requests, the data server should provide Web Map Tile Service (WMTS) to optimize data displaying on the web client side. Unlike printed maps which have a fixed scale, web maps can be zoomed to various scales and thus the map content needs to be updated simultaneously. The WMTS should consider this important factor. A common solution is to build an image pyramid. The image content will be generalized to different scales and form in each scale, the image will be tiled. When the user changes the zoom level of the web map on the client side, the image tiles of the corresponding scale will be rendered. Figure 36 shows the illustration of tiling images on different scales. A cache server should also be implemented to store frequently requested image data to ease the workload of the data server.

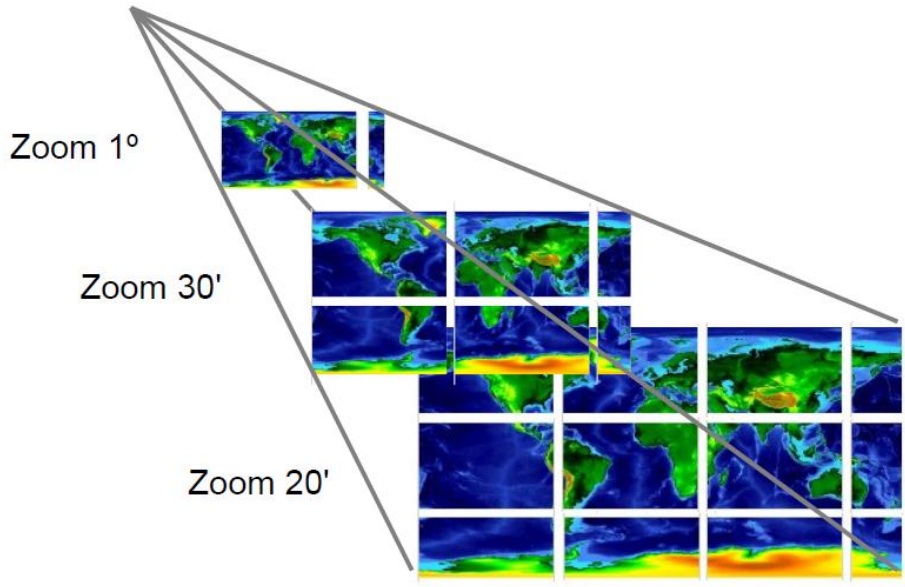


Figure 36: An illustration of WMTS and image pyramid (Masó et al., 2010)

4 Experiments, results and discussion

4.1 Colourization

The original DeOldify model was trained on the ImageNet dataset (Deng et al., 2009). To adapt this model to work better on colourizing CORONA satellite images, a retrain with the previously mentioned dataset is performed. The training is performed using an Nvidia A40 12-Q GPU with 12 GB memory on Windows 10 system on a virtual machine. All coding is achieved with python and fastai library.

The retraining starts with pretraining the generator with the CORONA image dataset. To avoid overfitting, data augmentation is implemented. Deep neural networks tend to overfit when trained with a relatively small amount of data (Shorten & Khoshgoftaar, 2019). The original DeOldify model was trained on the ImageNet dataset which consists of more than 10 million labelled images. In this study, however, the CORONA dataset only consists of 2000 labelled images. Thus, data augmentation is required to prevent the network from overfitting. Fastai library provides data augmentation functions to apply geometric and photometric transformations to training samples, including flipping, resizing, rotating, etc.

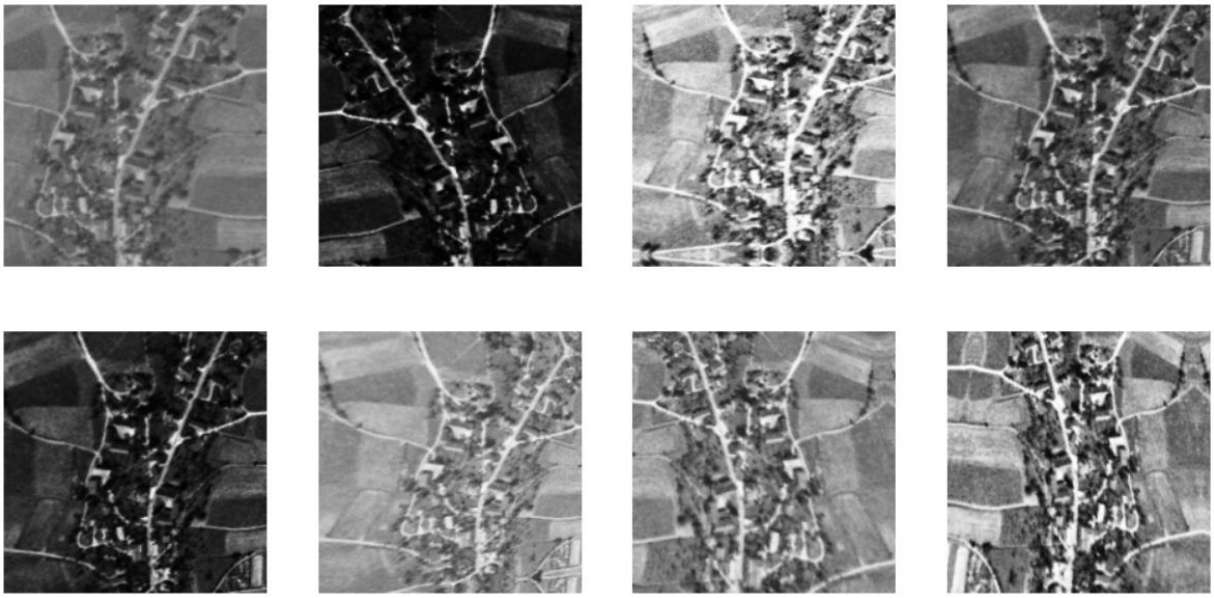


Figure 37: Apply data augmentation to one sample

The same transformations will be applied to the corresponding colour reference images.

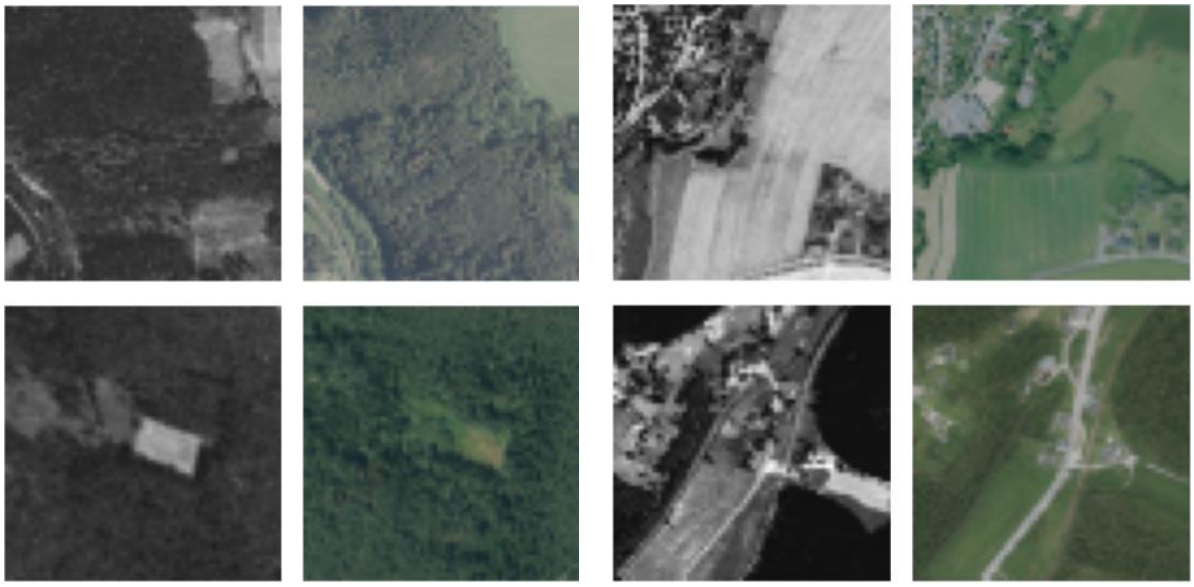


Figure 38: Training samples with label images

The pretraining of the generator is divided into 3 stages with each stage using images of a different resolution. The CORONA image dataset consists of images of 224x224 pixels. In the training process, these images will be resized to 64x64 pixels, 128x128 pixels and 192x192 pixels at each stage. The purpose is to learn image features at different scales. The batch size decreases from 88 to 22 and finally to 11 as the image size increases to avoid running out of GPU memory. In the first stage, the generator is first trained for some epochs while frozen to the last layer of the pretrained model (the downsampling path, a ResNet 34 model), and then unfrozen to train for more epochs. In the following two stages, the generator will both be trained while unfrozen. To freeze a network means not to update the weights in the selected layers or layer groups in the network while training. This is an important strategy while using a pretrained model to perform transfer learning (Weiss et al., 2016). The initial layers in a pretrained model mostly capture low-level features in the input data. Due to the usage of a large amount of data when training the pretrained network, these layers generally have a good performance in extracting these features. Thus, freezing these layers in the initial stage of training saves time and improves performance.

Initially, the stable mode was chosen for the pretraining of the generator, as according to the description of the original DeOldify model, the stable mode works better on landscape images and produces fewer glitches and artefacts. Different from the artistic mode, the downsampling path of the U-Net in the stable mode is based on a ResNet 101 model, which has a deeper architecture and takes more memory. Due to limit GPU memory, the initial batch size has to be greatly reduced. This has negatively affected the outcome of the training and thus the training was not continued.

After the trial with the stable mode, the artistic mode is selected for the retraining. In the original DeOldify model, the generator is pretrained for only one epoch with the ImageNet dataset at each training stage. Considering the difference in the amount of data, for each stage, the generator is pretrained with more epochs. In the first stage of the pretraining, 150 epochs are used while freezing the generator and 100 epochs while unfreezing the generator. The latter two stages of training both use 100 epochs of training. In total, 7 hours are spent on the pretraining of the generator. Once the pretraining for the generator finishes, the generator is saved and produces fake colour images for the critic's pretraining.

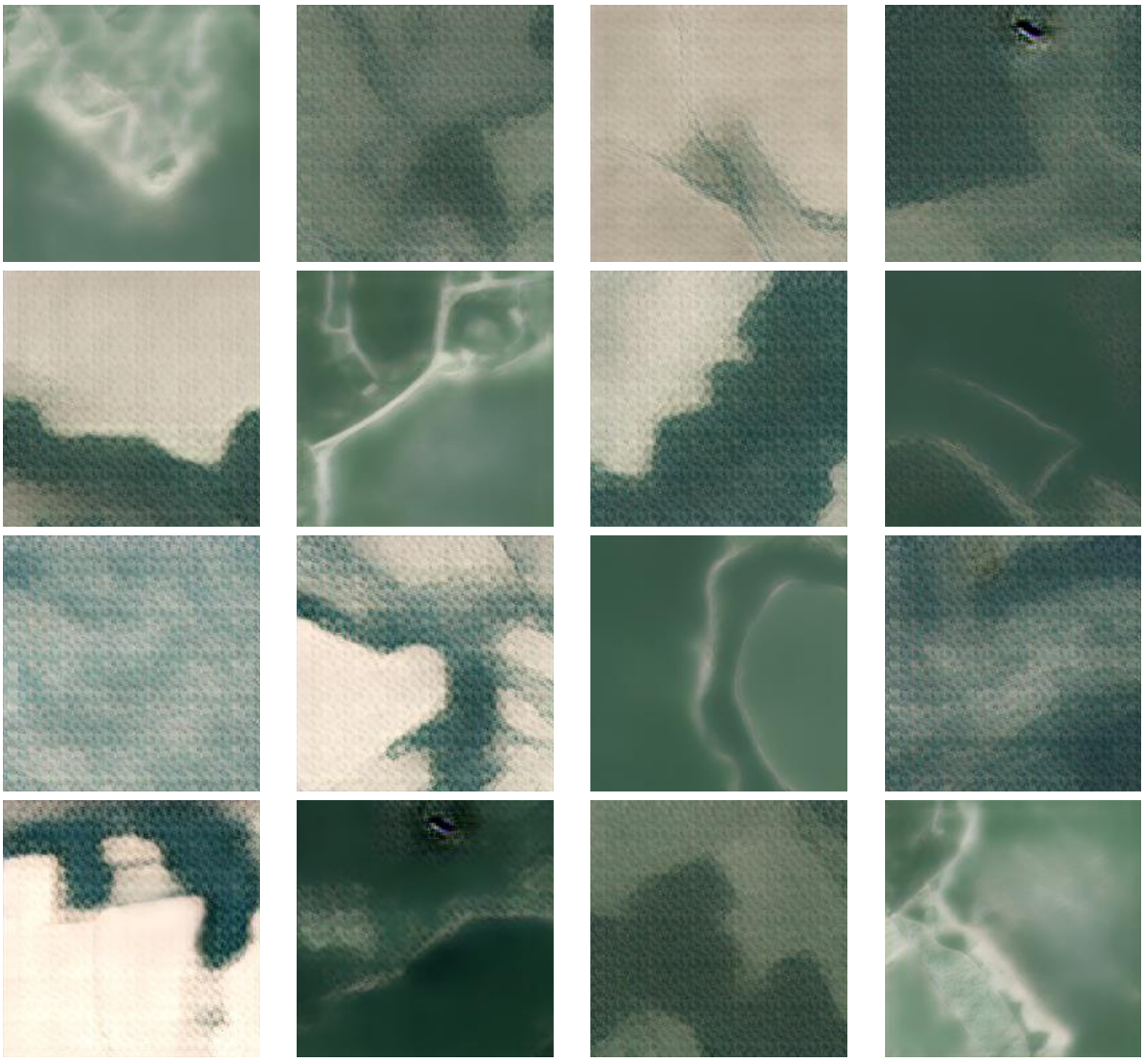


Figure 39: Samples from the images generated by the pretrained generator

Figure 39 shows some samples taken from the generated images by the generator. Edges of different land cover types can be roughly identified in images with forest and agricultural lands. But noticeable artefacts can also be identified in those images. Thus,

detailed features cannot be fully reconstructed. However, in images with regions with fewer textures or features and large homogeneous areas like water bodies, the reconstruction quality is significantly higher. And artefacts are almost fully eliminated.

The generated fake colour images will be used together with real colour images for the pretraining of the critic followed by the direct GAN training. The cycle of critic pretraining – direct GAN training will be performed for several rounds until an optimizing result is achieved. Data augmentation technique is also implemented.

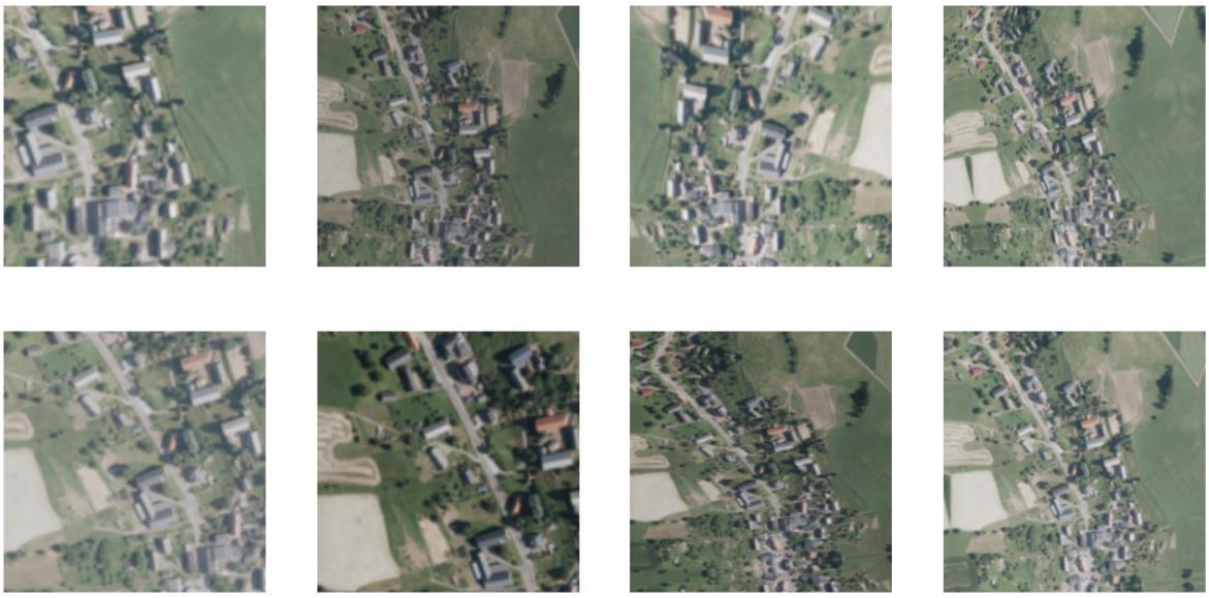


Figure 40: Apply data augmentation to training data for the critic

The labelled images as shown in Figure 41 will be used for the pretraining of the critic and GAN training.

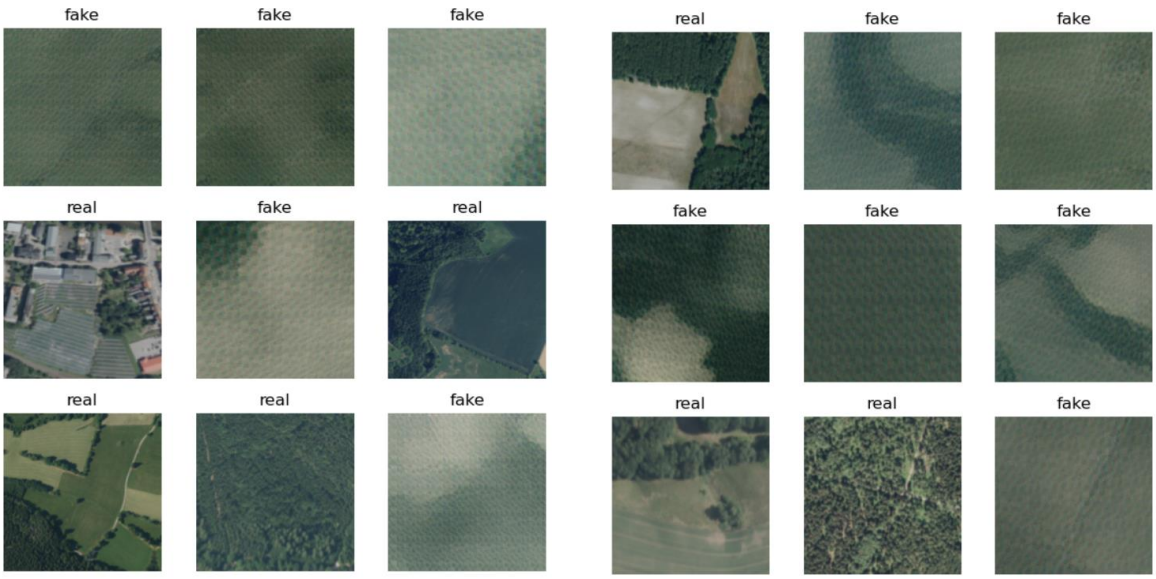


Figure 41: Training samples of the critic with labels

The pretraining of the critic is relatively easier compared to the pretraining of the generator. For each round of critic pretraining – direct GAN training, the critic will be pretrained for 30 epochs, and direct GAN training for 100 epochs. Each round of training takes about 5 hours. After each round, the generator will be saved and used for generating new fake colour images for the training in the next round. In total, 7 rounds of training are performed.

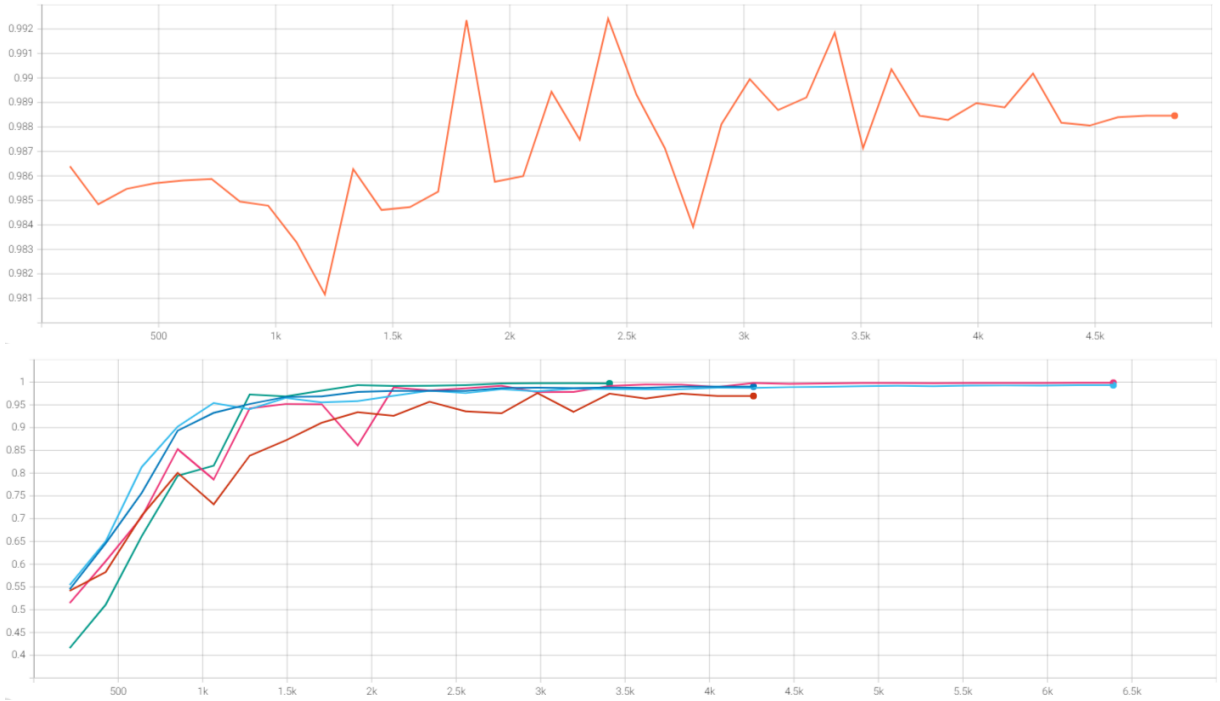


Figure 42: Accuracy plot of the first round of pretraining and following rounds of pretraining of the critic. It can be seen from the second plot that the accuracy quickly reaches nearly 1.0 after around 2000 iterations of training, which corresponds roughly to 9 epochs

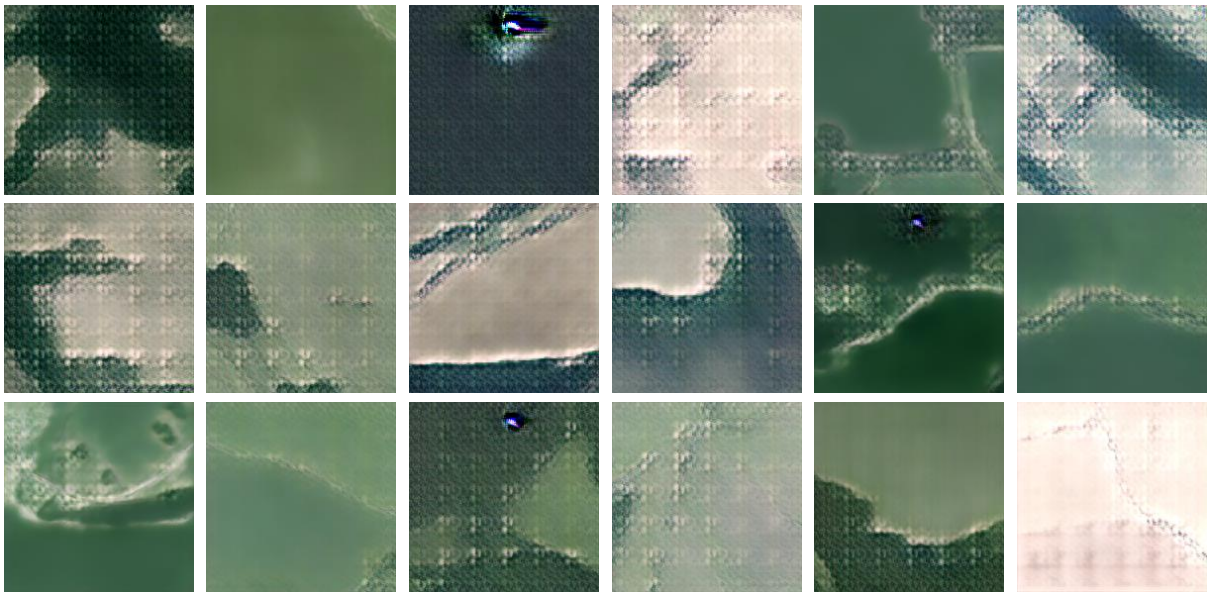


Figure 43: Colourized CORONA images after the first round of training

After the first round of training, as shown in Figure 43, the generated images have distinct reoccurring artefacts in regions with rich textures, especially in forest regions. Some error patterns are also identified in several images. However, the edges and boundaries between different regions in the same image become clearer. Colours also seem to be plausible.

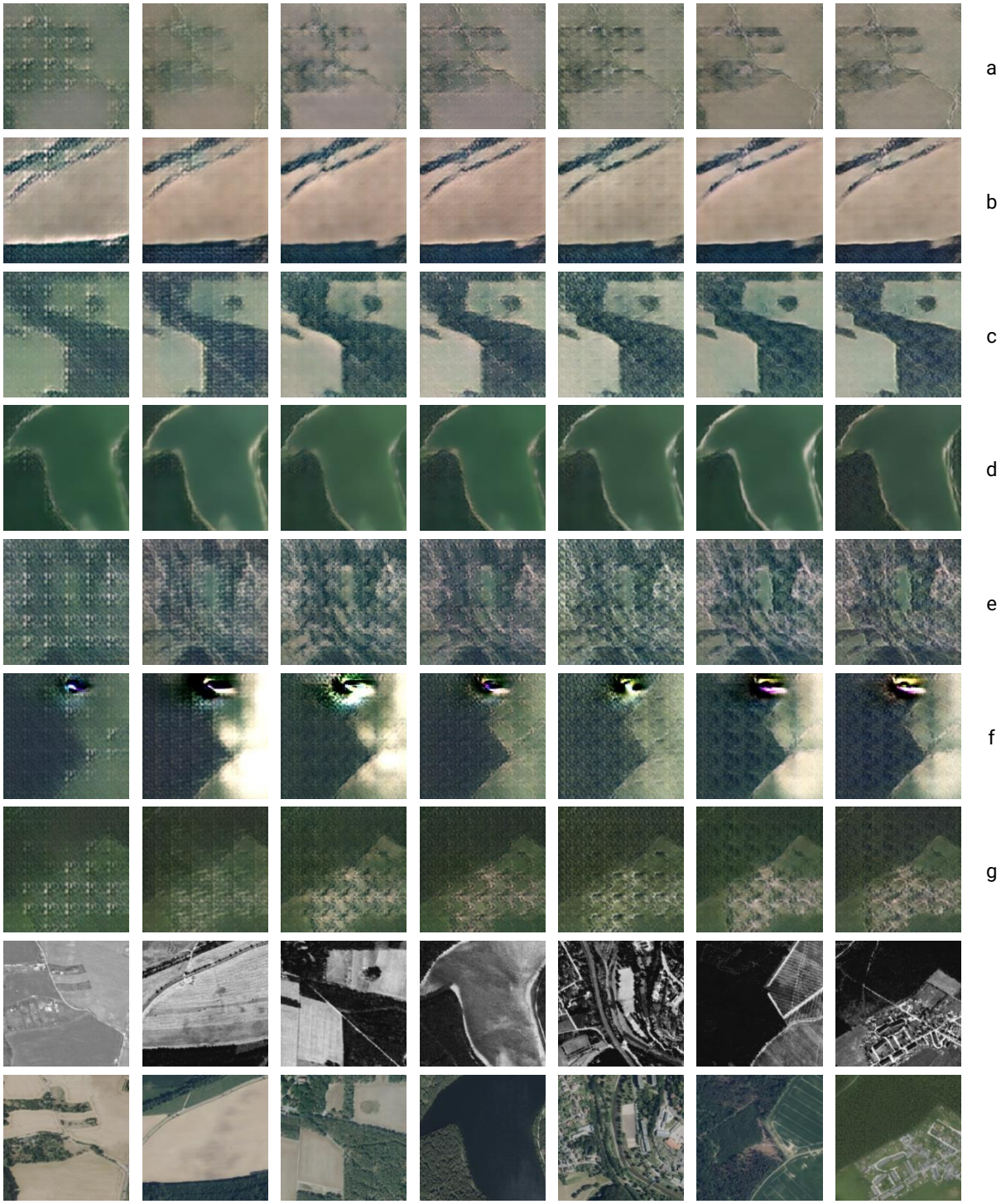


Figure 44: Selected sample results from 7 rounds of training and their corresponding originals and references

Figure 44 shows some selected sample results from the 7 rounds of training. For the first 7 rows, each row shows the results of the same image after each round of training. The last 2 rows show the corresponding original CORONA images and their colour reference images. It can be seen that artefacts in certain regions decrease or vanish as the training continues. These regions are mainly bare ground, agricultural land or water body, with large homogeneous areas but few changes in texture or patterns. On the other hand, artefacts in forest and urban land continue to exist as the training goes on, but with sight improvement which can be visually identified. Image c is a good example of how artefacts are dependent on different land cover types. It can also be seen that the output of the colourized CORONA images does not fully recreate the colours in the reference images. In image d, the water body has a brighter green colour in the colourized images while a darker green colour in the reference image.

The artefacts mostly centre on regions of rich texture changes or where the original CORONA image and colour reference image do not match well (for example, in urban areas where there are slight changes), and cannot be eliminated through more training. As shown in Figure 39, the generator only produces images with blurred content. Low-level features like edges and boundaries are identifiable, but high-level features like image details are primarily missing. This indicates that the pretrained generator failed to reconstruct the image details to a large extent. And this may have negatively affected the outcome of the NoGAN training, as the purpose of NoGAN training is to pretrain the generator in a conventional and more controllable way so that the generator can already produce high-quality images before the start of direct GAN training.

The original DeOldify model also provides the full weights of its pretrained generator. Because the original DeOldify model was trained on a massive dataset, the pretrained generator is expected to outperform the previously mentioned pretrained generator. Thus, the NoGAN training can start only from the pretraining of the critic followed by direct GAN training.

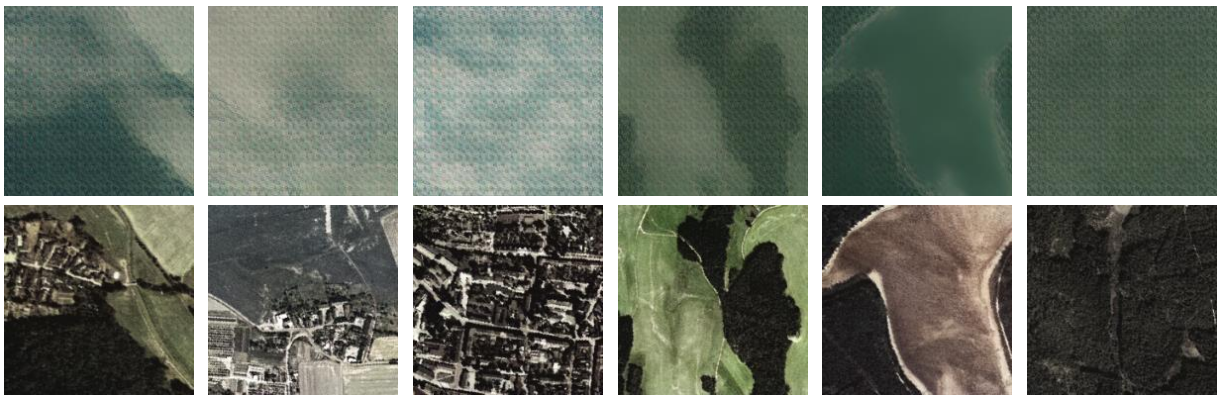


Figure 45: A comparison of the two generators' results

Figure 45 shows selected colourized results from the two generators after pretraining. The images in the first row are from the previously mentioned one, and the second row from the DeOldify model. The generator from the DeOldify model significantly outperforms the self-trained one in this study in terms of reconstructing image details. No noticeable artefacts are presented in the generated images. But they tend to have desaturated colours and thus a grey and brown tone in general.

The NoGAN training starts with the pretraining of the critic and uses the pretrained generator from the DeOldify model for GAN training. The parameters are set to the same as before. For each round of NoGAN training, the critic is pretrained for 30 epochs, followed by 100 epochs of GAN training. In total, the NoGAN training takes place for 3 rounds before no significant improvement is seen in generated images. For each epoch in the GAN training, the generator will be saved for inspection of the generated image qualities to sort out the model with the best performance.

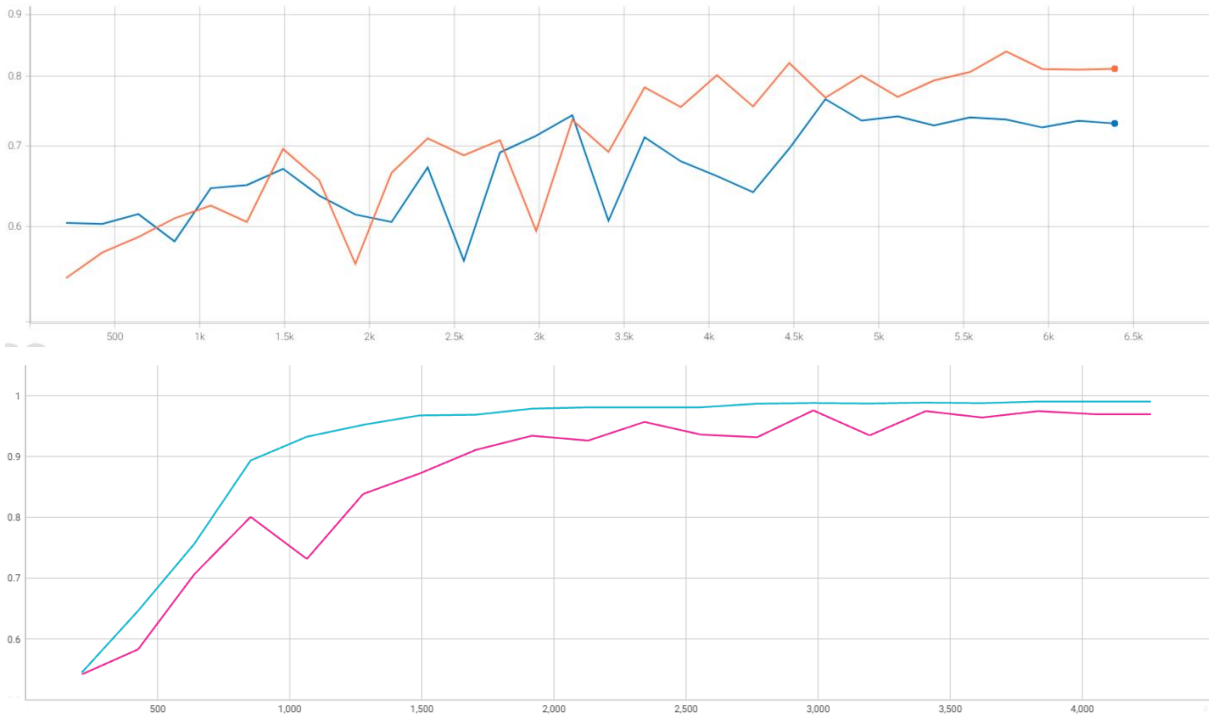


Figure 46: Comparison between the accuracy plots for the pretraining of the critic in the two trials

Figure 46 shows the two accuracy plots for the pretraining of the critic in the first and second rounds of the NoGAN training in the two trials. The upper plot is from the second trial, which uses the pretrained generator from the DeOldify model. It can be seen that within the same iterations, the critic has a lower accuracy in the second trial when using images generated from the generator from the DeOldify model. It is inferred that the fake colour images generated by the generators from the DeOldify model have a

higher quality. As a result, the critic has lower confidence when classifying them as fake images, thus leading to lower accuracy and a slower increasing rate.

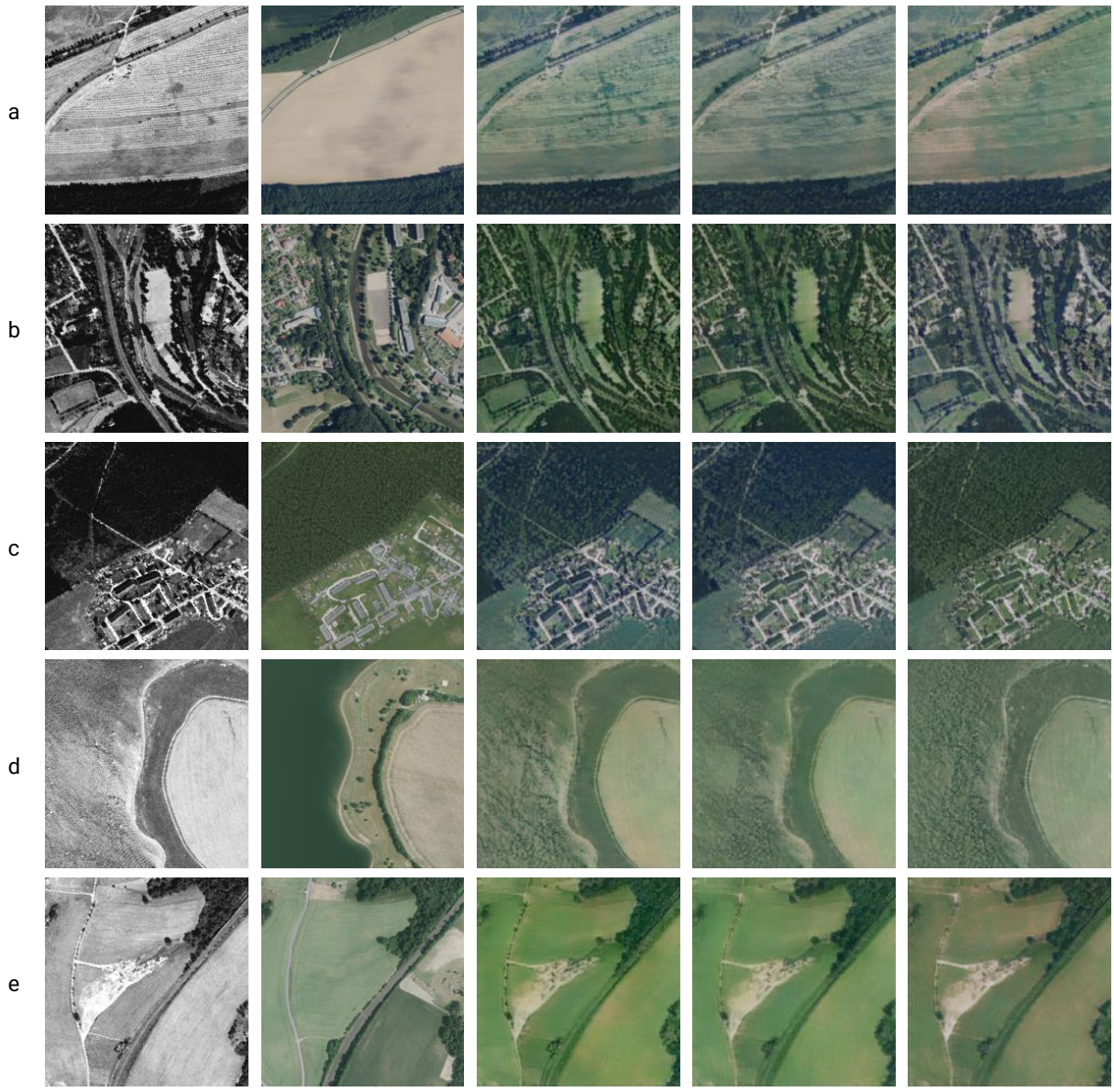


Figure 47: Selected samples from each round of NoGAN training

Figure 47 shows some selected samples from each round of NoGAN training together with the original CORONA images and the reference colour images. Compared with images generated in the first trial, these images in general have a higher level of detail and no significant artefacts or error patterns. Urban areas can be identified. Image features are mainly preserved. But colour-wise, these images tend to have a greener tone, especially in urban areas. Building roofs are mostly coloured green. It is inferred that

the bias of large areas of greenness contributed mostly by forest and agricultural land has resulted in this colourization trend.

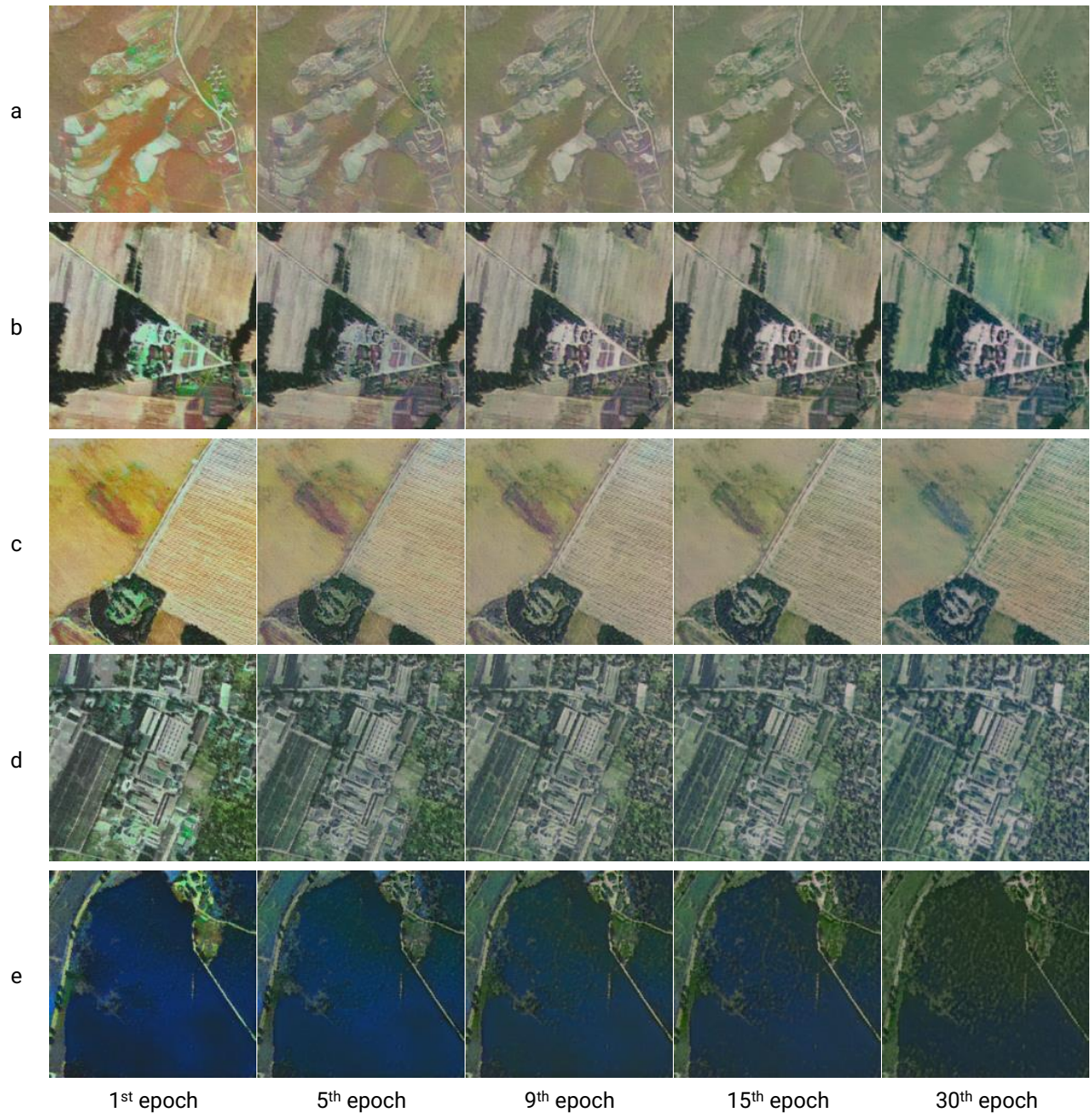


Figure 48: Samples from intermediate results after one epoch in one round of GAN training

Figure 48 shows results from the first round of GAN training after selected epochs. After the first epoch of training, the generated images are coloured in an unnatural way. The small residential areas in image b and image d appear to be light green after the first epoch. The forest area in image e shows a dark blue colour. As the training continues, the colours become more natural. But after the 30th epoch, unnatural green colours appear again in images b and c. The purpose of saving the generator model after each epoch of GAN training is to find the model with the best performance. After visual

comparison, the training ends after three rounds of NoGAN training. And the model after the 40th epoch in the second round is chosen as the final generator.

4.2 Evaluation

The evaluation is performed on selected test images. Test images are randomly selected image patches cropped from the greyscale CORONA images. But these images should contain as many land cover types as possible, including forest, agricultural land, bare ground, road, building, water body, etc.

Figure 49: Selected test images

As shown in Figure 49, 6 test images of size 560x560 pixels are chosen. The original DeOldify model uses a post-process for colourizing testing images. A parameter called *render factor* is specified by the users. All input test images are resized to the size of 16 * *render factor* before colourization. Then colourized images are resized back to the original size. This is equivalent to generating a low-resolution colour image. To restore the original image resolution, an image fusion is performed, converting the colour images to YUV colour space, and replacing the Y channel with the original greyscale images. To eliminate the possible effects of post-process on the evaluation, the test images are of the size 560x560 pixels, which is equivalent to setting the *render factor* to 35, the optimal value due to limited GPU memory. Thus, no resizing or post-process is required.

All test images are colourized and divided into four smaller patches for evaluation. The qualitative evaluation is a user study. Participants are asked if the displayed colour image looks natural to them. In total, 24 colour images will be shown. The 24 colour images consist of 9 real colour images, 9 images colourized by the retrained model (image (a) to (i) in Figure 50), and 6 images by the original DeOldify model (image (j) to (o) in Figure 51). These images are all derived from the 6 test images. The quantitative evaluation uses RMSE and PSNR as the metrics. A higher PSNR or lower RMSE value indicates better reconstructing quality compared to the original image. The hypothesis is that the retrained model should produce images with more plausible colours than the original DeOldify model, but not necessarily with better reconstructing quality.

(a): RMSE=36.3, PSNR=16.9, p=82.8% (b): RMSE=47.2, PSNR=14.6, p=82.8% (c): RMSE=32.9, PSNR=17.8, p=69.0%

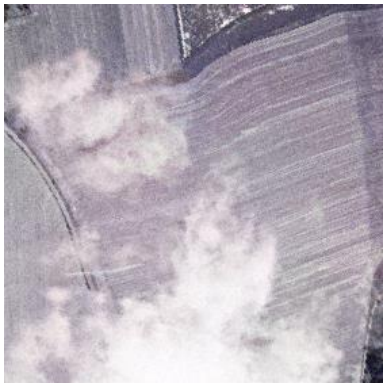
(d): RMSE=33.9, PSNR=17.5, p=58.6% (e): RMSE=36.7, PSNR=16.8, p=72.4% (f): RMSE=31.7, PSNR=18.1, p=41.4%

(g): RMSE=26.1, PSNR=19.8, p=96.6% (h): RMSE=33.7, PSNR=17.6, p=58.6% (i): RMSE=30.8, PSNR=18.4, p=69.0%

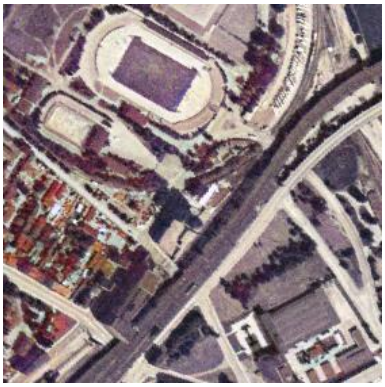
Figure 50: Evaluation results of the images colourized by the retrained model. RMSE is root mean square error, PSRN is peak signal-to-noise ratio, p indicates the percentage of participants thinking the image is natural

(j): RMSE=2.4, PSNR=40.7, p=27.6%

(k): RMSE=1.3, PSNR=46.1, p=20.7%



(l): RMSE=4.6, PSNR=34.9, p=44.8%



(m): RMSE=5.7, PSNR=33.1, p=24.1%

(n): RMSE=6.1, PSNR=32.5, p=13.8%

(o): RMSE=9.9, PSNR=28.2, p=31.0%

Figure 51: Evaluation results of the images colourized by the retrained model. RMSE is root mean square error, PSRN is peak signal-to-noise ratio, p indicates the percentage of participants thinking the image is natural

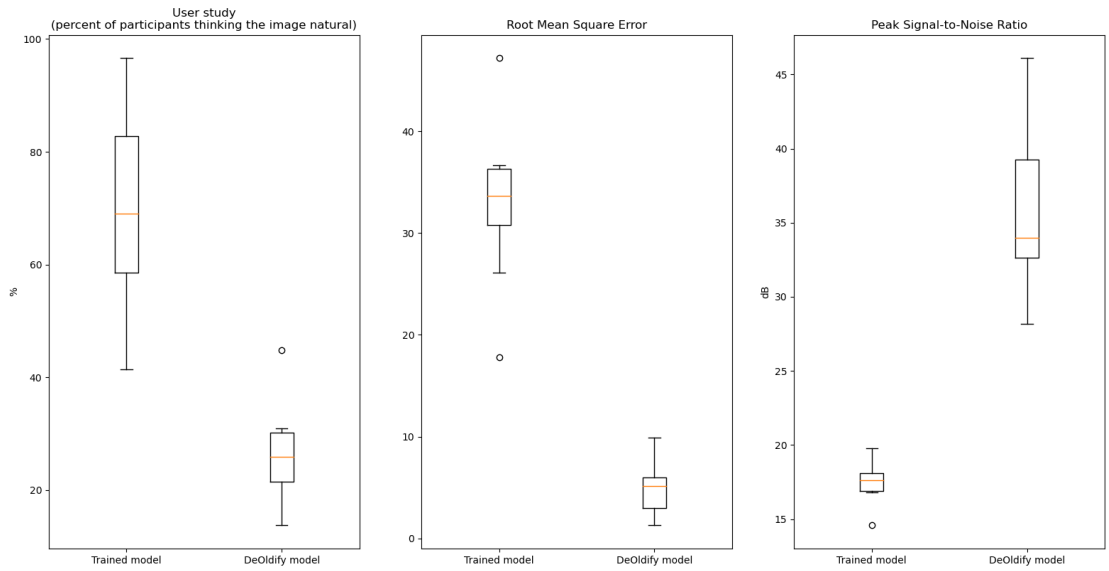


Figure 52: Boxplots for the three evaluation criterion. Each graph presents one of them. In each graph are two boxplots of the same criteria for images colourized by the retrained model and the original model respectively.

The results are shown in Figure 50, Figure 51 and Figure 52 in the form of boxplots, p indicates the per cent of participants that think the shown image looks natural. In total 29 participants take part in the user study. On average, 70.1% of the participants think images generated by the retrained model are real. Image (f) has the worst performance

among other images generated by the trained model. It is inferred that the unnatural yellow and green colours in the cloud have resulted in this. Image (i) also contains clouds, but compared with image (f), they are colourized more naturally. Image (d) and (h) also have a lower percentage. Buildings in these two images account for a large part and are easily identified, but they are mostly colourized in green. The unnaturalness can be clearly noticed. Although buildings also appear in image (g) and image (i), they are less noticeable. Thus, the green tone in these two images is not necessarily unnatural, especially for image g which has the best performance.

Overall, the images colourized by the retrained model have a significantly better performance on human eye colour perception. However, the DeOldify model generates images with much better numeric metrics, which indicates these images are very similar to the original images in terms of pixel-wise similarities and reconstructing quality. But they failed in terms of plausibility of colours. And this indicates that colourized images with better quantitative performance do not necessarily look natural and have plausible colours to human eyes.

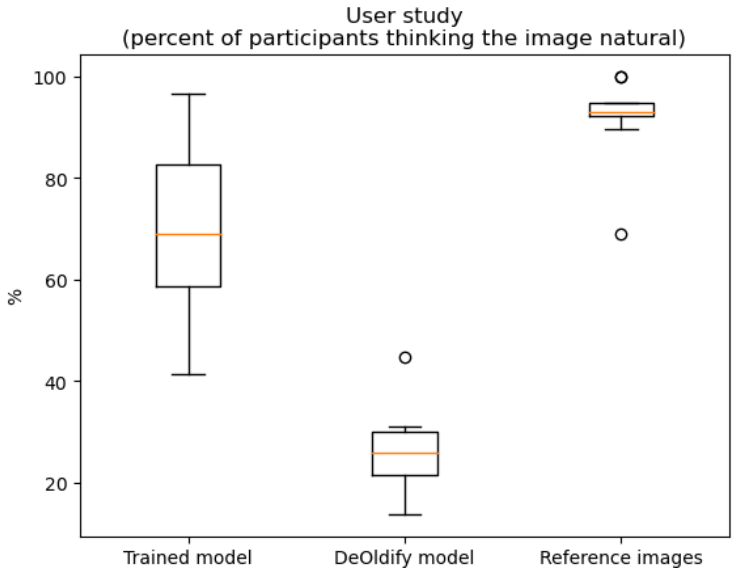


Figure 53: User study result comparison among all three types of images

Figure 53 shows the boxplot for all three types of images used in the user study, it is obvious that the colour reference images from the GeoSN digital orthophotos have the best performance. And this can be clearly seen in Figure 54, for images with buildings, the reference images have a significantly better colour quality and plausibility. The building roofs have more vivid colours than just green. This could be due to lacking enough images with buildings in the training dataset, and the inconsistency between CORONA images and reference images in buildings areas. Buildings may only exist in

one of the two images and the corresponding area in the other image becomes forest or other land cover types with mainly green colours. The dominance of green colours in the training dataset has negatively affected the colourization of the buildings.

Figure 54: Comparison between images colourized by trained model and references

Another data inconsistency may have resulted from the distortions in the CORONA images. The colour reference images are digital orthophotos, which have been orthorectified. However, the CORONA images are only georeferenced. Image distortions still exist. The CORONA images and the colour reference images may have pixel position shifts, and thus do not match pixel-wise. This could also have negatively affected the training and the final result.

4.3 Visualization

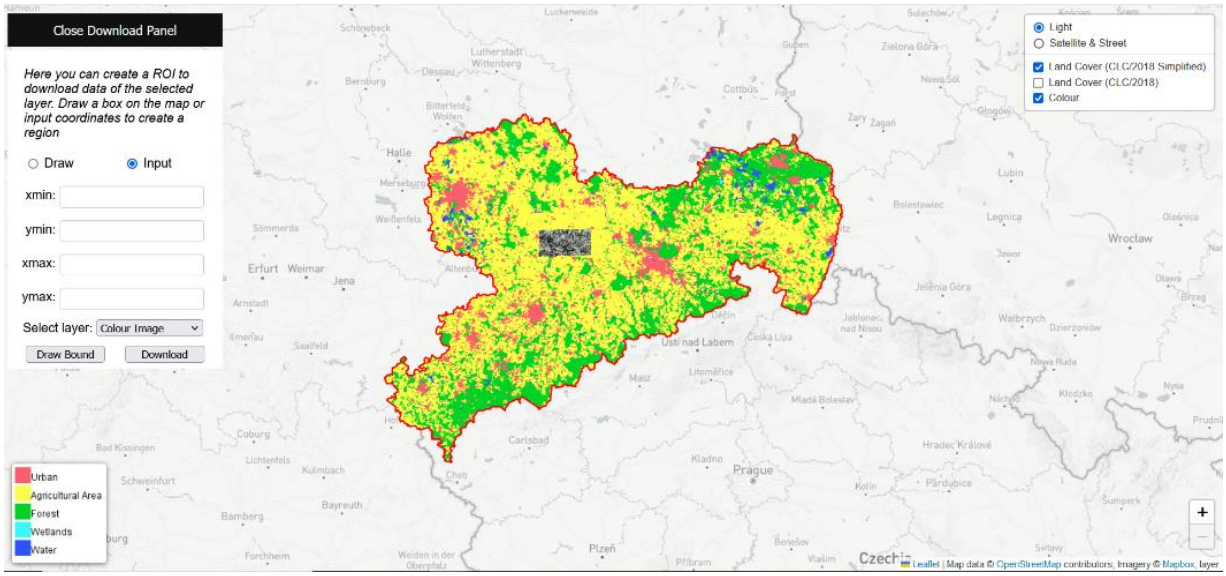


Figure 55: Interface of the web mapping application

In this study, Geoserver version 2.20.4 built on Windows 10 and Java 8, and Leaflet version 1.8.0 will be used to develop the application. Geoserver is an open-source server for handling geospatial data of various formats and implements OGC standards for data publishing, sharing and integration (Iacovella, 2017). Leaflet (Agafonkin, 2010) is an open-source JavaScript library for interactive map implementation on web browsers as well as mobile devices. It provides easy-to-use APIs to integrate data from Geoserver and can provide additional GIS functionalities through plugins. The web mapping application is developed using JavaScript, HTML and CSS

Visualization is in the form of a web mapping application, which consists of a frontend web map client built upon Leaflet, and a backend data server built upon Geoserver. The data server uses Geoserver to store data and publish data through WMTS for raster data and WFS for vector data. The web map client is built with Leaflet. Figure 55 shows the interface of the web map client.

In total six layers are added to this map through WMS. Leaflet provides direct APIs for implementing WMS. For easier usage, a third-party JavaScript library leaflet-geoserver-request is used. The Saxony state border is stored as a shapefile in the data server. But for a faster rendering, it is retrieved through WMS. Two base map layers are added through XYZ tile scheme and the data source comes from Mapbox. Three raster layers, the colourized CORONA image (randomly cropped from one of the film strips), a detailed land cover map according to CLC/2018, and a simplified land cover map, are added by WMTS from Geoserver. A layer control component provided by Leaflet is in the upper right corner, and the user can switch between the two base layers, and add or remove the three raster overlays.

On the upper left corner is the download control panel. Here user can define a rectangle region of interest by either manually inputting the coordinates of the four corners or drawing on the map through the drawing tool provided by Leaflet.

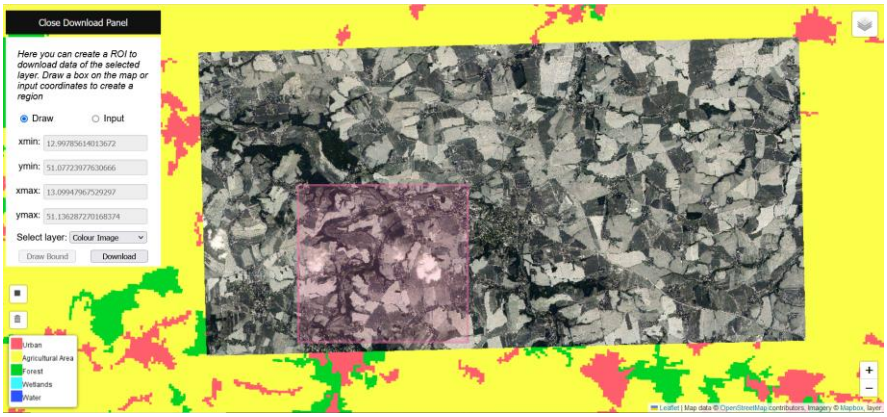


Figure 56: Define a rectangle ROI

As shown in Figure 56, the drawing tool is activated when the “Draw” mode is selected. The coordinates of the drawn rectangle will be automatically added to the text fields. When the mode is switched to “Input”, the user can click the “Draw Bound” button after setting all coordinates, and then the corresponding rectangle will be automatically drawn on the map. A drop-down is used for the users to select the layer they are interested in. By clicking the “Download” button, the download session will start.

Initially, the download function was implemented through WCS. The coordinates of the ROI will first be converted to match the projection of the image data. A WCS request is sent to the server with parameters including layer name, bounding box, resolution, coordinate system and output format. The result can be fetched by opening this URL.

This implementation is straightforward. By submitting an HTTP GET request, the result can also be easily fetched. But the response time can be longer if the requested data size is too large. Thus, the download function is improved and achieved by WPS. Geoserver provides WPS download plugins. Similarly, the web client side also sends an HTTP request to the server for downloading data. But the request is sent via the HTTP POST method and asynchronously, and the parameters are embedded in an XML file. The server receives the request and starts the download session. Meanwhile, the web map client can respond to other requests from the users while actively fetching the status of the download session. Once the session is finished, the client side will fetch the result. The time flow chart of the download session is shown in Figure 57.

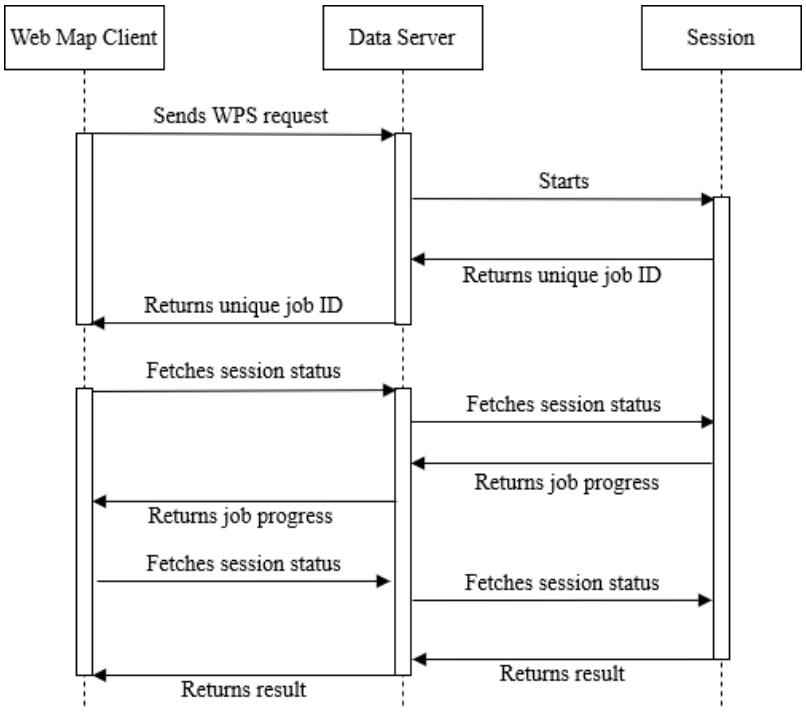


Figure 57: Time flow chart of the download session

The WPS implementation has a shorter response time than WCS. To download an image of 74.3Mb only takes less than 3 seconds with WPS, whereas with WCS it takes nearly 6 seconds. And with the asynchronous method, the web map client can handle other threads while waiting for the result when multiple users are requesting download sessions at the same time. However, this function can be further improved by setting up a temporary server to cache the requested layer. This way when the user sends a download request, the cache server will first try to respond and search the cached data and then either return the result or forward the request to the data server. Ideally, this could ease the data server and further improve the performance.

5 Conclusion

The overall research objective of this study is to colourize the grey scale CORONA images taken from the 1960s to 1970s in the region of Saxony and visualize the results. Specifically, this can be divided into three parts: to colourize the greyscale CORONA images, to evaluate the colourization results, and to visualize the results in the form of a web mapping application.

The colourization is achieved through the deep-learning-based method. The GAN-based DeOldify model, which is dedicated to colourizing black and white photos, is chosen. The original DeOldify model was trained on the massive ImageNet dataset. To adapt the model for the specific task of colourizing CORONA satellite images, a retraining of the DeOldify model on the CORONA image dataset is required. The dataset consists of CORONA images of selected study regions and their corresponding colour reference images. Retraining has been proven to be a good solution for the colourization problem. The use of the pretrained generator has greatly improved the outcome of the model, and the retraining on the CORONA dataset can lead to a better result than the original model. By using the transfer learning technique, the retraining process has greatly benefited from the existing model and saved a large amount of training time.

RMSE and PSNR are chosen as the quantitative measurements to evaluate the pixel-wise similarities and reconstructing quality, as well as a user study for qualitative evaluation. The results show that images with good numeric metrics do not necessarily have better visual quality. The images colourized by the retrained model significantly outperform the ones generated by the original DeOldify model in terms of naturalness and colour plausibility.

The web mapping application is built upon Geoserver as the backend data server and Leaflet for the frontend web map client. To handle possibly large size of image data, WMTS and cache service are used for a quicker response on the client side. Leaflet provides useful GIS functionalities for building an interactive web map interface. Users can browse and control the layers displayed on the map, and download interested dataset through WPS. To handle the possible situation that multiple users send download requests simultaneously, asynchronous threads are implemented to optimize efficiency.

Due to the time limit, some improvements can still be achieved in this study, and some future work can also be expected:

- Improve the data quality of the original CORONA images. If the digital elevation model of the study area is available, these images can be orthorectified to eliminate the distortions in the data. A more accurate match with the colour reference images can be expected after the orthorectification. Thus, the outcome of the retrained model could also improve.
- Select more training data with urban areas. The outcome of the retrained model does not perform well on colourizing urban areas, especially on building roofs. The situation could be improved with more urban areas included in the training data.
- Improve the download function in the web mapping application. A temporary cache server could be set up to ease the burden of the data server, and provide a quicker response to the client side.
- Add a swap tool in the web map interface. Users can use this swap tool to compare two image layers and get a more intuitive comparison.

Historical earth observation data is a valuable data source to study changes on the earth's surface. This study has contributed to the better exploitation of CORONA satellite images. Compared with greyscale images, colour images contain more spectral information. The additional information can be used to perform tasks like land cover classification and possibly lead to a better result. The web mapping application provides the user with an easy-to-use interface to interact with the colourized images. This can also benefit the implementation of cartographic techniques for visualizing deep learning results.

References

Agafonkin, V. (2010). *Leaflet*. https://leafletjs.com/

An, X., & Pellacini, F. (2008). Appprop: all-pairs appearance-space edit propagation. In *ACM SIGGRAPH 2008 papers* (pp. 1-9).

Andersen, G. (2006). How to detect desert trees using CORONA images: Discovering historical ecological data. *Journal of Arid environments*, *65*(3), 491-511.

Antic, J. (2018). *DeOldify*. https://github.com/jantic/DeOldify

Antic, J., Howard, J., & Manor, U. (2019). *Decrappification, DeOldification, and Super Resolution*. Retrieved 03.05.2019 from https://www.fast.ai/2019/05/03/decrappify/

Barik, R. K., Samaddar, A. B., & Samaddar, S. G. (2011). Service oriented architecture based SDI model for geographical indication web services. *International Journal of Computer Applications*, *25*(4), 42-49.

Bay, H., Tuytelaars, T., & Gool, L. V. (2006). Surf: Speeded up robust features. European conference on computer vision,

Cao, Y., Zhou, Z., Zhang, W., & Yu, Y. (2017). Unsupervised diverse colorization via generative adversarial networks. Joint European conference on machine learning and knowledge discovery in databases,

Cetin, M. (2009). A satellite based assessment of the impact of urban expansion around a lagoon. *Int. J. Environ. Sci. Technol.*, *6*, 579-590. https://doi.org/https://doi.org/10.1007/BF03326098

Charpiat, G., Hofmann, M., & Schölkopf, B. (2008). Automatic image colorization via multimodal predictions. European conference on computer vision,

Cheng, Z., Yang, Q., & Sheng, B. (2015). Deep colorization. Proceedings of the IEEE international conference on computer vision,

Chia, A. Y.-S., Zhuo, S., Gupta, R. K., Tai, Y.-W., Cho, S.-Y., Tan, P., & Lin, S. (2011). Semantic colorization with internet images. *ACM Transactions on Graphics (ToG)*, *30*(6), 1-8.

Choudhury, N. (2014). World wide web and its journey from web 1.0 to web 4.0. *International Journal of Computer Science and Information Technologies*, *5*(6), 8096-8100.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, *24*(5), 603-619.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. 2009 IEEE conference on computer vision and pattern recognition,

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Earth Resources Observation and Science (EROS) Center. (2008). *Declassified Intelligence Satellite Photographs* (No. 2008-3054). U.S. Geological Survey.

Earth Resources Observation and Science (EROS) Center. (2018). *USGS EROS Archive - Declassified Data - Declassified Satellite Imagery - 1* https://www.usgs.gov/centers/eros/science/usgs-eros-archive-declassified-

data-declassified-satellite-imagery-1?qt-science_center_objects=0#qt-science_center_objects

Gable, G. (2009). *Scanning Around With Gene: The Miracle of Photochrom*. Retrieved 13.11.2009 from https://creativepro.com/scanning-around-gene-miracle-photochrom/

Galiatsatos, N., Donoghue, D. N., & Philip, G. (2005). An evaluation of the stereoscopic capabilities of CORONA declassified spy satellite image data. 25th EARSeL symposium, workshop on 3D remote sensing, Porto, Portugal,

García, R., de Castro, J. P., Verdú, E., Verdú, M. J., & Regueras, L. M. (2012). Web map tile services for spatial data infrastructures: Management and optimization. *Cartography-A Tool for Spatial Analysis*, 26-48.

GeoSN. *Fachliche Details*. http://www.landesvermessung.sachsen.de/fachliche-details-5345.html

Geospatial World. (2009). *Recent developments in Internet GIS*. Retrieved 10-09-2009 from https://www.geospatialworld.net/article/recent-developments-in-internet-gis/

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, *27*.

Guo, Y., Zhou, M., Wang, Y., Wu, G., & Shibasaki, R. (2022). Learn to be Clear and Colorful: An End-to-end Network for Panchromatic Image Enhancement. *IEEE Geoscience and Remote Sensing Letters*.

Gupta, R. K., Chia, A. Y.-S., Rajan, D., Ng, E. S., & Zhiyong, H. (2012). Image colorization using similar images. Proceedings of the 20th ACM international conference on Multimedia,

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition,

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, *30*.

Huang, Y.-C., Tung, Y.-S., Chen, J.-C., Wang, S.-W., & Wu, J.-L. (2005). An adaptive edge detection based colorization algorithm and its applications. Proceedings of the 13th annual ACM international conference on Multimedia,

Iacovella, S. (2017). *GeoServer Beginner's Guide: Share Geospatial Data Using Open Source Standards*. Packt Publishing Ltd.

Iizuka, S., Simo-Serra, E., & Ishikawa, H. (2016). Let there be color! Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (ToG)*, *35*(4), 1-11.

Irony, R., Cohen-Or, D., & Lischinski, D. (2005). Colorization by Example. *Rendering techniques*, *29*, 201-210.

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE conference on computer vision and pattern recognition,

Jabs-Sobocińska, Z., Affek, A. N., Ewiak, I., & Nita, M. D. (2021). Mapping mature post-agricultural forests in the Polish eastern Carpathians with archival remote sensing data. *Remote Sensing*, *13*(10), 2018.

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. European conference on computer vision,

Larsson, G., Maire, M., & Shakhnarovich, G. (2016). Learning representations for automatic colorization. European conference on computer vision,

Levin, A., Lischinski, D., & Weiss, Y. (2004). Colorization using optimization. In *ACM SIGGRAPH 2004 Papers* (pp. 689-694).

Li, L., Lambin, E., Wu, W., & Servais, M. (2003). Land-cover changes in Tarim Basin (1964-2000): application of post-classification change detection technique. Ecosystems Dynamics, Ecosystem-Society Interactions, and Remote Sensing Applications for Semi-Arid and Arid Land,

Liu, X., Wan, L., Qu, Y., Wong, T.-T., Lin, S., Leung, C.-S., & Heng, P.-A. (2008). Intrinsic colorization. In *ACM SIGGRAPH Asia 2008 papers* (pp. 1-9).

Luan, Q., Wen, F., Cohen-Or, D., Liang, L., Xu, Y.-Q., & Shum, H.-Y. (2007). Natural Image Colorization. 18th Eurographics Symposium on Rendering,

Markle, W. (1984). The development and application of colorization. *SMPTE journal*, *93*(7), 632-635.

Masó, J., Pons, X., & Singh, R. (2010). OGC WMTS and OSGeo TMS standards: motivations, history and differences. *FOSS4G 2010*.

Neumann, A. (2012). Web Mapping and Web Cartography. In W. Kresse & D. M. Danko (Eds.), *Springer Handbook of Geographic Information* (pp. 273-287). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-72680-7_14

Oder, F. C. E., Fitzpatrick, J. C., & Worthman, P. E. (2013). *THE CORONA STORY*. CSNR Classic publication.

Peebles, C. (1997). *The Corona Project: America's First Spy Satellites*. Annapolis: Naval Institute Press.

Peng, Z.-R., & Tsou, M.-H. (2003). *Internet GIS: distributed geographic information services for the internet and wireless networks*. John Wiley & Sons.

Poterek, Q., Herrault, P.-A., Skupinski, G., & Sheeren, D. (2020). Deep learning for automatic colorization of legacy grayscale aerial photographs. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 2899-2915.

Putz, S. (1994). Interactive information services using World-Wide Web hypertext. *Computer Networks and ISDN Systems*, *27*(2), 273-280.

Qu, Y., Wong, T.-T., & Heng, P.-A. (2006). Manga colorization. *ACM Transactions on Graphics (ToG)*, *25*(3), 1214-1220.

Reinhard, E., Adhikhmin, M., Gooch, B., & Shirley, P. (2001). Color transfer between images. *IEEE Computer graphics and applications*, *21*(5), 34-41.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical image computing and computer-assisted intervention,

Ruderman, D. L., Cronin, T. W., & Chiao, C.-C. (1998). Statistics of cone responses to natural images: implications for visual coding. *JOSA A*, *15*(8), 2036-2045.

Sahin, K., & Gumusay, M. (2008). Service oriented architecture (SOA) based web services for geographic information systems. XXIst ISPRS Congress. Beijing,

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans. *Advances in neural information processing systems*, *29*.

Sample, J. T., & Ioup, E. (2010). *Tile-based geospatial information systems: principles and practices*. Springer.

Shahtahmassebi, A. R., Lin, Y., Lin, L., Atkinson, P. M., Moore, N., Wang, K., He, S., Huang, L., Wu, J., & Shen, Z. (2017). Reconstructing historical land cover type and complexity by synergistic use of Landsat Multispectral Scanner and CORONA. *Remote Sensing*, *9*(7), 682.

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. Proceedings of the IEEE conference on computer vision and pattern recognition,

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, *6*(1), 1-48.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

Sugawara, Y., Shiota, S., & Kiya, H. (2018). Super-resolution using convolutional neural networks without any checkerboard artifacts. 2018 25th IEEE International Conference on Image Processing (ICIP),

Tai, Y.-W., Jia, J., & Tang, C.-K. (2005). Local color transfer via probabilistic segmentation by expectation-maximization. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05),

Todo, H., Yatagawa, T., Sawayama, M., Dobashi, Y., & Kakimoto, M. (2019). Image-based translucency transfer through correlation analysis over multi-scale spatial color distribution. *The Visual Computer*, *35*, 811-822.

Tola, E., Lepetit, V., & Fua, P. (2008). A fast local descriptor for dense matching. 2008 IEEE conference on computer vision and pattern recognition,

Veenendaal, B., Brovelli, M. A., & Li, S. (2017). Review of web mapping: Eras, trends and directions. *ISPRS International Journal of Geo-Information*, *6*(10), 317.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of big data*, *3*(1), 1-40.

Weitz, A. (2020). *The Basics of Hand-Coloring Black-and-White Prints*. Retrieved 23.03.2020 from https://www.bhphotovideo.com/explora/photography/tips-and-solutions/the-basics-of-hand-coloring-black-and-white-prints%E2%80%82%E2%80%82%E2%80%82

Welsh, T., Ashikhmin, M., & Mueller, K. (2002). Transferring color to greyscale images. Proceedings of the 29th annual conference on Computer graphics and interactive techniques,

Wu, F., Dong, W., Kong, Y., Mei, X., Paul, J. C., & Zhang, X. (2013). Content - based colour transfer. Computer Graphics Forum,

Wu, M., Jin, X., Jiang, Q., Lee, S.-j., Liang, W., Lin, G., & Yao, S. (2021). Remote sensing image colorization using symmetrical multi-scale DCGAN in YUV color space. *The Visual Computer*, *37*(7), 1707-1729.

Xu, K., Li, Y., Ju, T., Hu, S.-M., & Liu, T.-Q. (2009). Efficient affinity-based edit propagation using kd tree. *ACM Transactions on Graphics (ToG)*, *28*(5), 1-6.

Yatziv, L., & Sapiro, G. (2006). Fast image and video colorization using chrominance blending. *IEEE transactions on image processing*, *15*(5), 1120-1129.

Žeger, I., Grgic, S., Vuković, J., & Šišul, G. (2021). Grayscale image colorization methods: Overview and evaluation. *IEEE Access*.

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019). Self-attention generative adversarial networks. International conference on machine learning,

Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful image colorization. European conference on computer vision,

Appendix

Link to all codes and data: https://drive.google.com/drive/folders/1kf0Bw5b-MkTAc-mJmej1jFF3Wuf_1h4LE?usp=sharing

Link to the user study form: https://forms.gle/NS5G36z6Uy97WZfA7