TÉCNICO LISBOA

Mapping Urban Areas Leveraging the Analysis of Ground-Level Imagery with Convolutional Neural Networks

Henrique Metelo Rita de Almeida

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Doutor Bruno Emanuel da Graça Martins
Prof. Doutor Jacinto Paulo Simões Estima

Examination Committee

Chairperson: Prof. Doutor Luís Manuel Antunes Veiga
Supervisor: Prof. Doutor Bruno Emanuel da Graça Martins
Members of the Committee: Prof. Doutor João Carlos Gomes Moura Pires

January 2021

Acknowledgements

First of all, I would like to thank Professor Bruno Emanuel da Graça Martins and Professor Jacinto Paulo Simões Estima for their commitment in providing me support for the last year and a half, especially during the unusual times of the last few months. Their knowledge and suggestions were essential for the development and completion of this work.

I would also like to thank my family, especially my parents and brother, not only for providing me the motivation and the chance to study at Instituto Superior Tećnico, but also trying to help and review my work in every way that was possible to them.

I would also like to acknowledge the importance of the work developed by computer science community in general. All the help and tools that are publicly available online were a huge contribution to this dissertation and enabled me to test different approaches to my problems without having to build implementations from the ground-up.

Finally, I would like to thank my friends and professors who provided me the motivation and guidance that I needed during these challenging, but also amazing, 5 last years of my academic life at Instituto Superior Técnico.

Henrique Metelo Rita de Almeida

For my family,

Resumo

Os avanços tecnológicos verificados em dispositivos móveis têm vindo a permitir um acesso cada vez mais fácil a ferramentas fotográficas, como a câmara de um smartphone. Este facto, juntamente com os avanços nas tecnologias de conectividade, criou a possibilidade de uma partilha simples de fotografias na web. Em plataformas como o Flickr ou Geograph, é possível encontrar-se uma quantidade quase infinita de imagens partilhadas, frequentemente acompanhadas de informação de geolocalização. Estas constituem fontes de informação das quais se podem extrair detalhes sobre a área onde foram obtidas. Esta dissertação tem como objetivo a exploração da informação disponível nestas colecções de modo a criar dois tipos de mapeamento: o uso dado ao terreno e a beleza cénica. Nestes mapeamentos, as imagens ao nível do solo podem fazer a diferença na obtenção de resultados mais precisos. O procedimento desenvolvido consiste numa recolha de fotografias sobre uma região, seguido da geração de mapeamentos do terreno, através do uso de redes convolucionais e recorrentes, de modo a combinar a informação proveniente de sequências fotográficas. O método apresentado possibilita o uso de um modelo de mapeamento automático alternativo ao mapeamento manual do terreno, mesmo quando aplicado em áreas onde existe uma variação significativa na densidade de fotografias disponibilizadas por zona. No final deste trabalho, são apresentados dois mapeamentos, um do uso do terreno e outro de beleza cénica, que permitem exemplificar de que maneira o uso de redes convolucionais e recorrentes pode ser aproveitado para a análise e extração de informação de sequências de imagens.

Abstract

The technological advancements in mobile devices have allowed people to easily take pictures and share them on the web. Within platforms like Flickr and Geograph.uk, we find an almost infinite number of community-shared pictures, often containing the location where they were taken, which might provide information about the surrounding area. This paper proposes an approach that aims to exploit such information to tackle two mapping tasks, land-use and scenic-beauty mapping, which are two examples for which ground-level photos can be the key to achieve accurate results. The procedure explored in this work consists of collecting data for a study region, which comprises the city of London, and generating mappings for both tasks by making use of convolutional and recurrent neural networks. This work presents the results obtained by aggregating images into sequences, which are initially processed by a convolutional neural network and transformed into sequences of features that are then fed to recurrent neural units, in order to combine their information and extract projections for either land-use or scenic-beauty mapping. The proposed method presents itself as an automated alternative to hand-annotations for terrain mapping, and as an improvement over the analysis of individual images. In the end, the result of applying the methods proposed in this work is presented in the form of two maps for the study region, one for land-use classes and one for scenic beauty, which were automatically generated based on the information extracted from the available set of community-shared photographs.

Palavras Chave
Keywords

Palavras Chave

Dados Geo-Espaciais

Informação Partilhada em Redes Sociais

Mapeamento de Uso da Terra

Mapeamento de Beleza cénica

Redes Convolucionais Profundas

Redes Recorrentes

Keywords

Geospatial Data

Community-Shared Information

Land-Use Mapping

Scenic-Beauty Mapping

Deep Convolutional Neural Networks

Recurrent Neural Networks

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The recent growth in the number of community-shared geotagged ground-level images, mainly due to the increasing use of smartphones for photographic purposes, has provided a new source of information which may help identifying geographical features of the surroundings in which each photograph was taken. Several projects, such as the Flickr Creative Commons Dataset[1] and the Geograph.uk[2], were created with the intent of collecting and aggregating large sets of georeferenced pictures which might prove useful for a vast array of tasks, such as terrain mapping tasks. Several recent works, such as in Leung and Newsam (2010) and Zhu et al. (2019) have explored this idea by employing state-of-the-art image analysis methods to create a variety of mappings, in which ground-level georeferenced photographs have proven to be advantageous when compared with the more traditional use of top-down satellite images.

Previous studies (Newsam and Leung, 2019) have focused on collecting and analyzing images one by one, and then trying to produce mappings by associating a class to each single image. Some studies (Srivastava et al., 2020) have considered an alternative approach in which the information of several images could be combined in order to improve the extracted mapping. The aim of this work is to build a model based on a similar alternative, where instead of making an analysis image by image, the model generates a representative sequence of photographs for each unit of area and then extracts a mapping by leveraging state-of-the-art deep neural networks. To evaluate the quality and performance of the envisioned model, I will focus on two tasks, generating land use and scenic beauty mappings.

1.2 Thesis Statement

In urban areas, mapping land use is critical for local governments that intend to execute smart and informed city planning initiatives in order to provide sustainable growth of the urban fabric. Land usage stands for the use that is given by humans to a certain occupied area. Some examples of land use categories are Residential areas, Industrial areas, Sports facilities, or Commercial zones.

The task of creating these mappings usually cannot be accomplished using satellite level images

[1] http://www.flickr.com/explore
[2] http://www.geograph.org.uk/

as, although certain areas can be analysed relying on top-down images (e.g. sports facilities by identifying soccer fields), most of times, certain details that might be the key to obtain such mappings can only be captured from a ground-level perspective, such as building facades (e.g. store fronts). As a consequence, the production of this type of maps usually requires hand-annotations, obtained by survey-based methods, which imply significant manual labour. Therefore, it is not possible to keep these maps up to date on a regular basis. Some efforts (Qui et al., 2019) were made to create new approaches that introduced an automation component, using inference strategies, but this did not remove the need for a set of manual annotations. This work mainly focuses on leveraging community-shared ground-level images to fully automatize this mapping by making use of state-of-the-art neural network architectures.

Other concept which is hard to quantify from a top-down view is the scenic beauty of a place. A region with a high scenicness score consists in a area which possesses aesthetically pleasing natural features, such as mountain ranges or natural parks. Obtaining an absolute scenic score that can depict a region is usually a hard task to achieve, as it might vary according to one's definition of scenic beauty. In addition, this measure is also affected by a variety of factors, such as the field of view (i.e. area captured by the camera lens), the flora or the human presence (e.g. man made buildings), which can only be determined by taking into consideration a ground-level view of the area.

The main dataset explored in this work also features scenic beauty scores for a range of images, which were obtained by averaging scores attributed by three or more people. As a consequence, this work also explores an automated approach to the task of mapping scenic beauty, based on the model obtained for land-use mapping. This approach will consider the same testing region of the previous task and will also explore the relation between land-use classes and scenic beauty score.

1.3 Contributions

The work developed for this thesis aims to provide the following contributions:

- The application and evaluation of the most recent state-of-art Convolutional Neural Network(CNN) architectures to the task of land-use mapping based on image-to-image analysis, using data collected from both *Geograph* and the *Urban Atlas*. Recently presented architectures, such as the EfficientNet(Tan and Le, 2019), have provided the possibility to train models with increased accuracy, while using a lower number of tuning parameters. As such, it would be interesting to find how these novel architectures would behave in the tasks considered for this thesis. In addition, a variety of new methods, such as the AugMix(Hendrycks et al., 2019) augmentation technique, are tested in order to evaluate its impact on the mapping results.
- The proposal a mapping technique, based on the analysis of sequences of images by making use of recurrent neural networks. In this work, this technique is studied and applied to the previously mentioned tasks of land-use and scenic beauty mapping.

- Evaluation of the proposed mapping technique on the task of land-use mapping, by comparing the obtained results with those of alternative methods proposed in previous works. The models for sequence analysis are built on top of the CNN architecture that provided the most promising results for image-to-image mapping tasks. The impact of other factors, such as the variation in the size of the sequence of images, are also presented in this work. Overall, this evaluation suggests that the proposed method presents equal or slightly improved results when compared with previously tested method when applied to land-use mapping.
- The application of the same convolutional and recurrent neural network architectures to the task of scenic beauty mapping, based on the images collected for the task of land-use mapping and by using the information provided in the Scenic-Or-Not game[3], which provides average scenicness scores for photographs collected from Geograph, based on reviews given by at least 3 different people.

Overall, the main goal of this work is to present a model that can successfully tackle a terrain mapping task in an automated way based only on previously available knowledge and community-shared ground-level photographs from the study region. The desired final output of the model consists in a raster map that can effectively portrait a study region based on a set of classes or scores that are gathered based on the information extracted from images. The data and code used for the development of this work are available at `github.com/Henrique97/ThesisFinalWork`.

1.4 Organization of the Document

This dissertation is divided into sections which are presented in the following order: In Section 2, I provide a brief explanation about some basic concepts related to deep neural network models, which play a key role in this work, along with several works that address the use of community-shared ground-level images to produce mappings for a variety of tasks, such as land use, land cover or scenic beauty. Section 3 presents a definition of the objectives of the proposed work, along with a short description of the methodology and data sources that I have used in this work. In Section 4, I present some details of the used methodology along with the results for both of the explored tasks. Finally, in Section 5 I present an overview of the work as well as possible paths to further expand the proposed approach for automated terrain mapping.

[3] `http://scenicornot.datasciencelab.co.uk/`

2 Concepts and Related Work

2.1 Fundamental Concepts

This section describes fundamental concepts for the understanding of the proposed work. It is divided into two parts with the intent of providing some basics about neural networks, and then proceeding to expand on the subject of Convolutional Neural Networks (CNNs) for image classification.

2.1.1 Supervised Learning with Deep Neural Networks

The elementary unit which composes neural networks is commonly referred to as the perceptron. A perceptron is a unit capable of generating an output from a given set of inputs by obtaining an intermediate value resulting from a linear combination of the inputs, and then passing it through a non-linear function that is usually referred to as an activation function. In practice, for a set of inputs $\boldsymbol{x} = \{x_1, ..., x_n\}$, we can combine them according to the formula:

$$p = \sum_{i=1}^{n} w_i \times x_i + b \tag{2.1}$$

In the previous equation, $\boldsymbol{w} = \{w_1, ..., w_n\}$ is used for representing the weights associated with each input, and b represents a bias term. For a binary classification program we can then obtain a map to two classes, using $\mathrm{sign}(.)$ as the activation function, thus having Å· as {0,1}, as follows:

$$\hat{y} = \mathrm{sign}(p) \tag{2.2}$$

Taking into consideration general classification problems, a single perceptron by itself can only be considered a linear classifier. This derives from the fact that there is only one output which results from a linear combination of the inputs, ruling out the possibility to depict functions that are not linearly separable, such as the XOR function. However, joining multiple perceptrons into a network of connected layers not only allows us to obtain better classification results, by increasing the processing power of the model, but also allows us to find solutions for problems that are not linearly separable. This is the idea behind neural networks, which base their behaviour on a simple model of a network of biological neurons, with the neurons being represented by the perceptrons that in turn are disposed in an interconnected structure, corresponding to a network. A simple neural network structure that is commonly used

is the Multi-Layer Perceptron (MLP), composed by several connected layers of perceptrons, each of them representing a processing level in the network's hierarchy. The first and last layers are responsible for receiving the input and presenting the output, respectively, with the middle layers, also named hidden layers, performing the processing. By adding hidden layers we can increase the processing power of the network. A MLP containing a large number of hidden layers is often referred to as a deep neural network.

In order to use a neural network to obtain a correct mapping between an input and output for a specific problem, the network needs to be trained, which requires tuning all the network parameters (i.e. all weights and biases). The most common approach to address this issue, considering the supervised learning paradigm, which makes use of ground truth samples for which we already know the correct mapping, is known as backpropagation. Backpropagation begins by calculating the difference between the expected and the predicted outputs, according to the network's current state, and uses that information to tune the network's parameters according to the obtained error. The difference between the predictions and the ground-truth is assessed through a loss function (e.g. the least mean square error in the case of regression problems) to obtain an error associated with the network's predictions. After this step, this error is used to update the weights and biases w_{ij}, with i representing a node with j input weights, which can be achieved by calculating the gradient of the error function with respect to the parameters of the network, and using the following rule to update the weights:

$$w_{ij}^{t+1} = w_{ij}^{t} + \eta \frac{\partial E}{\partial w_{ij}} \tag{2.3}$$

In the previous equation, t represents the current training iteration, η the learning rate of the network and w_{ij}^{t+1} is the updated value of the weight w_{ij}.

By using the generalized delta rule, it is possible to send the error backwards to previous layers, which allows separate gradient calculations for the hidden and output layers. Looking at the output layer, the computations are simple and can be done by using the following formulas:

$$\frac{\partial E}{\partial w_{ij}} = \delta_i x_j \quad \text{with} \quad \delta_i = (y_i - \hat{y_i}) \times f_i'(p_i) \tag{2.4}$$

In the previous equation, y_i and $\hat{y_i}$ are the expected and predicted output for node i, respectively, f' is the derivative of the activation function chosen for the node, and p_i is the linear combination of the inputs for the node i, similar to what happens in Equation 2.1. As we can see in the formulas, if the activation function is differentiable we can use backpropagation to deal with learning.

For a hidden layer k, the gradient can be calculated by applying the chain rule, using the gradients of the subsequent layers according to the formula:

$$\delta_i^k = f'(p_i^k) \sum_j w_{ij}^{k+1} \times \delta_j^{k+1} \tag{2.5}$$

It is important to be aware of some problems that might affect a network's performance when using backpropagation on deep networks. One example, known as the gradient instability problem, consists on vanishing or exploding gradients that can be expected for some activation functions when considering networks with more than a couple of hidden layers. This problem leads to difficulties when tuning some hidden layers, especially the first ones. An illustrative example is the use of a sigmoid activation function, in which the derivative is always smaller than 0.25. In a network with 6 layers this would result in a maximum update factor of 0.00024 for the first layer, which would make it impossible to effectively train the network. To deal with this issue, other activation functions can be used in these cases, such as the ReLU function Hahnloser et al. (2000) which has its derivative between 0 and 1.

Due to the benefits of using gradient descent to iteratively tune a network by reducing the loss function, backpropagation is the most widely used technique to train neural networks. To improve the performance of the training algorithm, several optimizations were built on top of the traditional application of the gradient descent method Ruder (2016). These optimizations aim to help achieving faster convergence of the network's parameters, reducing the computational cost. One simple optimization for gradient-based learning is to avoid the traditional gradient calculation over all of the training dataset. This is achieved by employing a variation of the standard algorithm such as Stochastic Gradient Descent (SGD), which performs one parameter update per sample, or mini-batch gradient descent, which divides the dataset into smaller batches and makes an update for each one of them. On top of these techniques, other types of methods aim to further improve performance, while also dealing with some of the issues when using gradient-based learning. One of the most commonly used methods to help parameters to converge faster is known as Adaptive Moment Estimation (ADAM). In essence, ADAM joins the benefits of other optimimizers, such as AdagGrad and RMSprop, into one algorithm and helps achieving faster convergence at a lower cost by adapting the learning rate individually for each parameter, according to its importance. To achieve this, ADAM stores a chunk of the last training steps to generate running averages of past gradients and their squares, and uses this information to choose an appropriate learning rate for each weight parameter. Considering $E[\delta]_t$ and $E[\delta^2]_t$ the running averages for the last gradients and the corresponding squares, respectively, we can make a parameter update according to the following formula:

$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{\hat{E}[\delta^2]_t + \epsilon}} \times \hat{E}[\delta] \tag{2.6}$$

In the previous equation, $\hat{E}[\delta^2]$ and $\hat{E}[\delta]$ are the result of a bias-corrected initialization for the running averages, since these estimates begin with a value of zero and can remain with a small value for a long time, which would slow down the parameter updates.

2.1.2 Convolutional Neural Networks for Image Classification

Convolutional Neural Networks (CNNs) are a type of neural network specialized in dealing with high-dimensional data (e.g., images). Although the principles behind this type of network are the same as those from standard neural networks, CNNs can provide dimensional filters that allows us to identify patterns in an image, while keeping the number of learnable parameters significantly low when compared to standard neural networks. To achieve this, CNNs employ the use of convolution operations to deal with the dimensionality problem. The convolutional operation makes use of filters, which are composed of trainable weights, and convolves them with the input. To give a clearer picture of the operation, consider a 2x2 filter f and a 4x4 input matrix x shown below:

$$f = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix} \quad x = \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{bmatrix}$$

The convolution operation would begin by flipping the filter along its height and width, resulting in:

$$f' = \begin{bmatrix} 4 & 1 \\ 3 & 2 \end{bmatrix}$$

With the filter flipped, considering a sliding window in the matrix with the size of the filter, the next step is to slide the filter along the matrix's width and height, calculating a weighted sum of the resulting values inside each window. The above procedure will result in a 3×3 matrix where for each cell y_i, we can obtain its value by using the formula:

$$y_i = \sum_j x_j . w_j \tag{2.7}$$

In the previous equation, x_j are the values for the input cells inside the sliding window and w_j are the associated weights for filter f. An illustration for this operation is presented in Figure 2.1. The resulting 2D output is the following:

$$y = \begin{bmatrix} 10 & 13 & 20 \\ 20 & 23 & 30 \\ 30 & 33 & 40 \end{bmatrix}$$

In a convolution operation, there are also some hyper-parameters that can be tuned in order to control the dimensionality and other properties of the output, according to the problem specificities. The

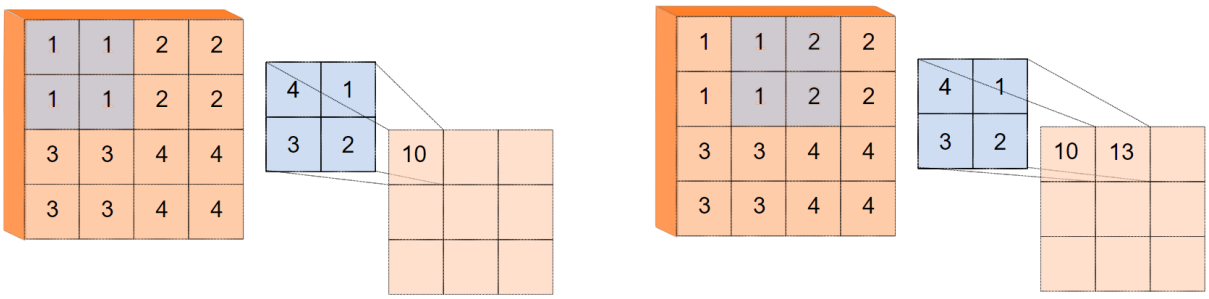

Figure 2.1: First two steps of a convolution operation

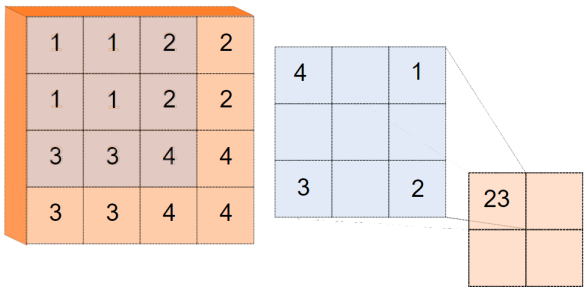

Figure 2.2: A convolution operation with dilation. By having a dilation factor of 1 it is possible to use a 2×2 filter to cover a 3×3 region, though some information will be lost due to the ignore units of input. The work of Zhou et al. (2015) shows that these convolutions allow a $3.6\times$ decrease in the number of parameters with only a 1% decrease in accuracy in a classification problem.

stride is a hyper-parameter that can be used to modify the jump size when sliding the filter through the input, allowing us to decrease the dimensionality of the output, while the padding parameter is used to add outer layers of zeros that work as a way to increase the output dimensionality. Both the decrease and increase in dimensionality have several use cases such as object recognition and image de-noising, respectively.

Considering the context of image classification, the idea behind the use of CNNs is to begin by dividing an image into small groups of pixels, where local features can be extracted, and work the way up until image wide features are identified, such as objects. To do this, CNNs usually aggregate a vast number of different types of layers. Some of the most simple layers which are commonly found in a network's architecture are detailed next.

The main layers in a CNN are known as convolutional layers and are responsible by applying the aforementioned convolutional operations. These layers can have a vast number of variable characteristics such as filter size, stride, padding, as well as a dilation parameter. This last feature is related to the fact that most convolutional layers resort to small sized filters to avoid having a large number of trainable parameters, which can be damaging for problems requiring pixel-wise dense predictions (e.g. segmentation). By using a type of convolution called dilated convolution, which employs a dilation, a layer is capable of increasing the spatial range of a filter, in relation to the input, without increasing the number of trainable weights, as shown in Figure 2.2.

Another common type of layer, found in CNNs, are the pooling layers. These layers are able to

down-sample the input by applying a pooling function, such as the maximum or the average function. To exemplify, we can consider a pooling layer which applies a pooling function to a 2×2 region for a 4×4 input and with stride 2. We can then picture the input as an aggregation of four 2×2 regions and for each one of them we will calculate the average of all its values. The result will be a 2×2 output where each cell contains the average value for each region of the input. These layers provide not only a more compact representation, but also some invariance that helps with the detection of similar objects or forms with different scales, translations, or positions.

Finally, in the last part of a CNN, fully connected layers are usually present. Fully connected layers are similar to the weight layers present in the MLPs and are usually used to combine the features of the previous layers and map them to the desired classes that should be output.

It is important to keep in mind that both fully connected layers and convolutional layers usually use non-linear activation functions, which allows the network to learn nonlinear mappings. These functions are usually differentiable to enable gradient-based learning.

Looking at CNN training, it is important to notice that this type of networks tend to overfit, i.e. although a network achieves good performance on the training set, it is not able to generalize for data outside of this set. To deal with this issue we can employ several regularization techniques, such as data augmentation, where we increase the size of the dataset by applying rotation, cropping, flipping and other operations to some samples, in order to create newer ones. Other examples of regularization ideas are the dropout-connect technique, which deactivates some of the network connections to force all neurons to contribute to the output, and the batch-normalization method. Looking at batch-normalization in more detail, during the training step, this method can be applied to reduce shifts in values in hidden units (covariance shifts) that result from the need to adapt to changes that occurred in the distribution of the activations from previous layers. By introducing a way to normalize the input at the layer level, batch-normalization allows hidden units to converge faster, by reducing the need to adapt to the changes in the output of previous layers, at each learning step.

To give a more clear picture of a CNN architecture, we can look at one of the most simple and basic CNN, the LeNet (LeCun et al., 1998). In Figure 2.3 we can observe one variation of the LeNet architecture, known as LeNet-5. This type of network has a total of 5 weight layers and its architecture consists in two sets of one convolutional layer followed by one max-pooling layer, and one set of a convolutional layer feeding two fully-connected layers.

2.1.3 Recurrent Neural Networks for Sequence Classification

In some cases, neural networks benefit of having loops in their structure, in which an output from a neuron can be passed back to the network to be used in later processing stages, as shown in Figure 2.4. This technique can lead to improvements in a network's performance for situations where data presents sequentiality, such as a time or spatial sequence, as it allows a set of inputs to be processed, while

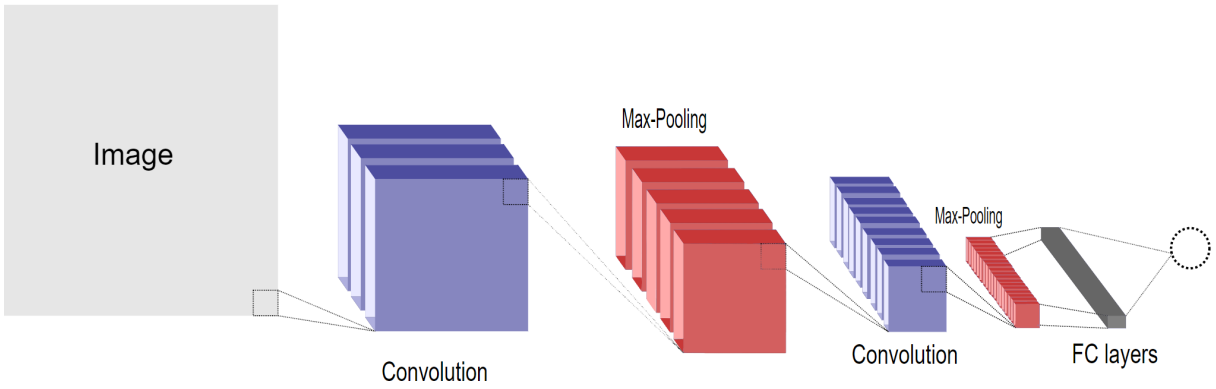

Figure 2.3: LeNet-5 structure. Based on an illustration from LeCun et al. (1998).

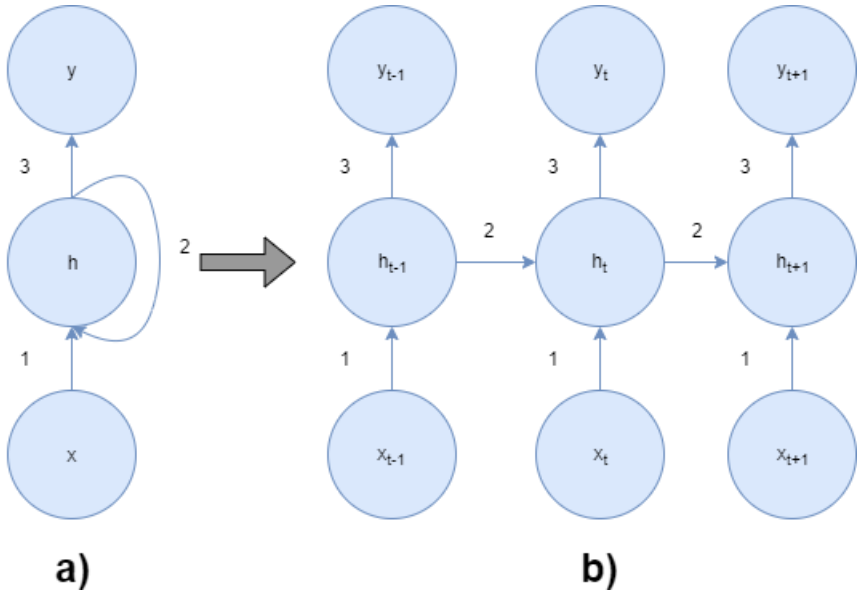

Figure 2.4: A RNN feedback loop is presented in **a)**. In **b)**, this same loop can be seen unfolded for different timesteps. Based on an illustration from Khan et al. (2018).

considering the previous states. Networks that present this behaviour are known as Recurrent Neural Networks (RNN).

The capacity of storing short term information allows this type of networks to present a certain kind of features that differentiate them from common networks.

Firstly, a RNN is capable of processing variable size inputs, due to the way it generates an output by combining both input and the result from its previous state. This behaviour can be very valuable for certain cases, such as video processing, where it enables the network to adapt its number of layers according to the number of frames that need to be processed.

Another feature of RNNs is the fact that it can provide an output with variable length. This is possible due the capacity of the network of storing information from previous inputs in a hidden state(h_t), which allows us to shape the output(y_t) not only from the input, but also from the stored information from previous states(h_{t-1}). This can be applied when trying to make predictions based on sequences, such as predicting the next state of a sequence.

Finally, it is important to mention that these networks are able to be efficiently trained due to having the ability of sharing parameters for several processing steps. For each cycle in an RNN, the parameters are kept the same, which largely reduces the number of variables that need to be tuned on the learning steps. However, due to the dependency on a hidden state to act as a memory, this type of networks can have difficulties storing knowledge from more than a few states away, losing information as it gets older. To deal with this issue, some optimizations were introduced to allow networks to have improved memory capacity, such as the gating mechanism found in Long Short-Term Memory networks (LSTM).

A LSTM network is a variation of a RNN in which each processing unit is divided in 3 gates, which perform different tasks, as shown in 2.5. Unlike common RNNs, which only contain a hidden state(h_t), each unit in an LSTM, also known as a cell, receives and keeps an extra information vector, known as cell state(c_t), which is updated in an additive rather than a multiplicative manner, which allows the network to keep a more stable memory, capable of holding long-term information. The first gate, known as forget gate, is responsible for verifying what information should be kept from the previous cell state(c_{t-1}), given the hidden state from the previous layer(h_{t-1}) and the new input(x_t), and using a linear combination of these variables as an input to a sigmoid function(σ). Next, the input gate filters the information of the input and previous hidden state that is meaningful for the output by using a sigmoid function to assess which values should be updated, and a hyperbolic tangent function(tahn) to obtain a vectorized representation that can be added to the previous cell state. The results of these operations is combined with the filtered cell state obtained in the forget gate. It is important to notice that the different types of operations used in the cell state update (i.e. sum) and hidden state update (i.e. multiplication) are the key to the long short-term memory of LSTMs as it allows them to keep both a long-term and short-term memory information flow, respectively. Finally, the output gate is responsible for combining the information from the input and previous hidden state, filtered by a sigmoid function, with the memory stored in the cell state, to produce a new hidden state and output. Overall, a LSTM provides the ability to smooth the loss of memory by introducing a cell state which is updated in an additive manner, allowing the information to be lost at a much slower rate. Due to this behaviour, LSTMs avoid the vanishing gradients issue, which heavily affects RNNs when the sequences provided as input are too long, though it remains vulnerable to the exploding gradients issue.

2.2 Related Work

In this section I present some work that has already been done in this area. In section 2.2.1, I discuss several approaches related to leveraging ground-level imagery to map urban areas, while in section 2.2.2, I present some advanced and state-of-the-art neural models which will be used for feature extraction in this work.

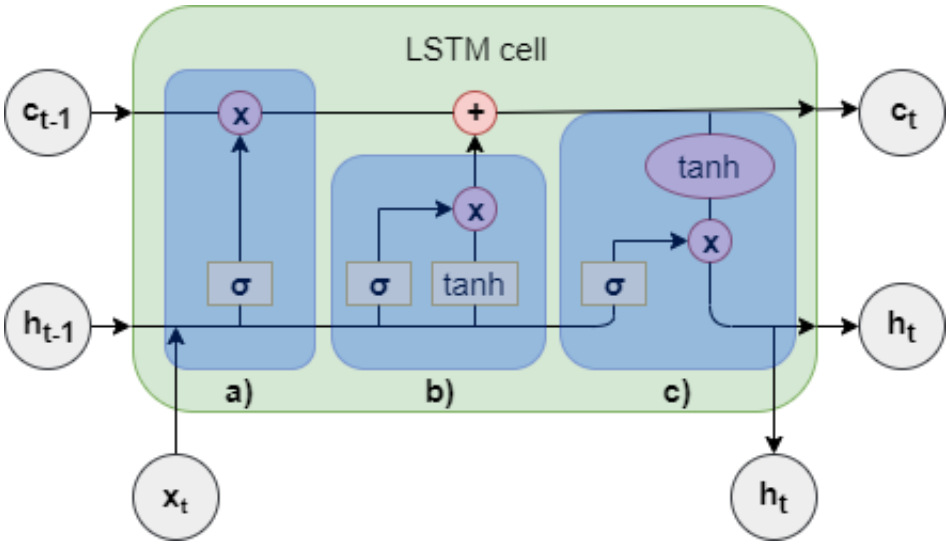

Figure 2.5: An LSTM cell. The three gates inside each cell are highlighted in blue: **a)** Forget Gate; **b)** Input Gate; **c)** Output Gate. The red circles represent update operations, by using $+$ for addition and $\times$ for multiplication operations. The sigmoid and hyperbolic tangent functions are represented by σ and $tanh$, respectively. x_t represents the input at timestep t, while the cell state and hidden state are represented for both the previous timestep (c_{t-1} and h_{t-1}) and the current timestep(c_t and h_t).

2.2.1 Mapping Urban Areas Leveraging Ground-Level Imagery

Nowadays, most of the images used for terrain mapping and classification tasks are usually top-down images, such as satellite imagery, as they are usually easier and faster to obtain, and cover big areas. However, this type of pictures are not appropriate for some types of mappings, such as the ones which are proposed in this work, due to the fact that they do not allow to capture certain obscured details, such as building facades. As such, we can leverage ground-level imagery to achieve better results in those tasks. Several works, have already used this type of images as a starting point to develop mappings and classification techniques, usually recurring to community-shared images which also provide the location where they were taken.

Figure 2.6: Examples of edges extracted from a develop area (left) and an undeveloped area (right). From Newsam and Leung (2019).

2.2.1.1 Previous Approaches Leveraging Feature Extraction

The first type of approach used to leverage this data had its focus on feature extraction from community-based images, in order to use it as a volunteer geographic information source (Newsam and Leung, 2019). This work focused on dealing with the noisy and complex data that exists in these types of photos, by applying a concept named proximate sensing (Leung and Newsam, 2010), which in practice implies extracting low, medium and high level features, based on the visual content of the photos, and use them along with simple classifiers, like a Support-Vector Machine (SVM), to map the terrain in a set of classes.

The first application of proximate sensing suggested in this work is the use of ground-level imagery to map land cover, which translates into identifying the physical materials that are present in the images, such as grass, trees or concrete structures. This is a type of mapping that is easy to accomplish using proximate sensing, as it only requires a low-level analysis. Although this mapping can also be accomplished using satellite images, using proximate sensing allows us to obtain better results for cases in which some of the visual features cannot be observed from an overhead perspective, usually due to occlusion.

Looking at a more concrete example, we can apply proximate sensing to differentiate between developed and underdeveloped regions. The authors have proposed this work as a proof of concept, as previous works using overhead images can provide a ground truth for this problem, allowing to focus the work on improving the effectiveness of proximate sensing. To achieve the desired mapping, the work proposes to focus on the orientation of edges that can be extracted from the images, as developed scenes tend to have a larger number of horizontal and vertical edges, that are associated with buildings and other types of structures, as shown in Figure 2.6. By using a supervised learning approach we can then feed these features to a SVM classifier with radial basis functions, and obtain a label for each image in the dataset. Finally, by dividing an area into a set of tiles and using a binary classifier to aggregate the results from the several images in each tile, we can obtain a land cover map.

After some optimizations, such as the use of a weekly-supervised learning framework and the use of a more trustworthy dataset, the authors obtained a correct classification rate of 74.7% for an area of 100*100 km of Great Britain, which includes the London metropolitan area.

Although the results are promising, the work proposes further optimizations, such as the use of bigger, but less trustworthy datasets, by including some filtering of pictures that should not contribute to the final result, such as portraits.

The second application of proximate sensing, which is also more in line with the proposed work, relies on ground-level imagery to map land use. This has a greater interest than land cover mapping, as it is usually harder to rely on satellite images to complete this task. This time, a map of a university campus was used, as it provided a ground truth map. The same methodology was used and the area was divided into tiles. To achieve the land use mapping, the work proposed the use of a weekly-supervised training set, as it provided better results in the previous task than the fully supervised approach. The algorithm

consisted in extracting a bag of visual words (Sivic and Zisserman, 2003) from each image, which were then passed to a one-versus-all SVM architecture to associate a class to each image. Finally, the images were aggregated according to the tiles they belong to, and the majority class was assigned to the tile. Although this experience has achieved some fairly good results, there were situations where there was clearly room for improvement.

The work also exemplifies the use of proximate sensing for public sentiment mapping, which translates into mapping sentiments, such as the scenicness of a location, and finalizes by suggesting some improvement to the proposed methodologies, such as the combined use of overhead and ground-level images.

2.2.1.2 Previous Approaches Leveraging Convolutional Neural Networks

Although feature extraction can be applied in certain contexts, it usually lacks in the capability to generalize for other contexts and, in order to obtain improved results, it implies a lot of work cleaning and sorting the data. To overcome this issue, we can use Convolutional Neural Networks (CNNs).

In Zhu et al. (2019), this idea is explored by making use of a novel CNN architecture to map 45 land usage classes for the city of San Francisco, resorting to large image sharing services, such as Flickr. The proposed CNN consists of a two stream architecture, where one stream is responsible for object recognition, with the other being used for scene recognition, as several tests pointed out that the combination of both streams presented better results than just employing a simple one stream architecture.

One of the main challenges faced by the authors was the lack of a ground truth map for fine-grained mapping, which implied producing one by themselves. The idea to tackle this problem consists of using the Google Places indexed points of interest to extract labels for certain locations that could then be used to derive land use for a certain area. Although the achieved results were acceptable, this technique added some degree of error to the ground-truth maps, due to wrong or misunderstood labels and due to regions with conflicting labels, where different land use classes can be encountered (i.e. there is no class for which the prediction score stands out from the rest), which can lead to regions where there are conflicting labels. Looking at the use of CNNs for this task, the authors opted to employ a Resnet101 (He et al., 2015), which provided the best relation between accuracy and efficiency, and used a batch size of 256 of images coming from Flickr and Google platforms. Due to the noise in the available data, a method named adaptive online learning can also be employed to obtain better performance. This technique works as an alternative to a manual cleaning process, which would be impossible to apply due to the size of the data, and tries to filter out the samples that the model struggles to classify. The idea behind this behaviour is that the information in these ambiguous samples will most likely confuse the model, rather than help its convergence, so the obtained results should not used for backpropagation. To establish which samples should be used, considering a problem with n classes and the prediction scores $y_i = [y_{i1}, y_{i2}, ..., y_{in}]$ obtained for sample i, we compute the following formula to calculate the

Table 2.1: Results for a 45-way land use classification problem. Based in Zhu et al. (2019).

	accuracy	precision	recall	f1 score
SIFT	29.16	4.56	12.85	3.37
SIFT + FVE	31.20	5.01	13.67	3.67
ResNet101 (Pre-Trained)	37.87	7.92	18.98	5.59
ResNet101 (Fine-Tuned)	43.90	10.57	21.67	7.10
ResNet101 (Object)	46.73	12.30	25.41	8.29
ResNet101 (Two-stream)	49.54	14.21	29.06	9.54

probability of a sample being used:

$$p_i = \max(0, 2 - \exp|\max(y_i) - \bar{y_i}|) \tag{2.8}$$

In the above formula, p_i represents the probability of a sample i being used for training, $max(y_i)$ identifies the top score obtained in the set of predictions (i.e. label that would be attributed to sample i), and $\bar{y_i}$ is the average score obtained, considering the predictions for all n classes.

We can then establish a threshold, for which the authors opted for 0.5, and avoid using samples with a lower probability than the defined value. This method has proven to be beneficial, allowing a boost in the accuracy of the model.

To test the effectiveness of this model, the results of performing end-to-end learning using the proposed CNN architecture can be compared with more common approaches of performing feature extraction and classification separately. In this work, the authors began by extracting hand-crafted features using Scale-Invariant Feature Transform (SIFT). SIFT consists in a keypoint detector that allows the extraction of features by identifying objects present in a picture, even when these present different rotations, scales or positions. The obtained representations were then encoded as Bag of Visual Words vectors (using k-means or Fisher Vector Encoding). Deep learning features were also extracted from the last fully-connected layer of a pre-trained ResNet101 CNN (i.e. previously trained in the ImageNet database). As presented in Table 2.1 we can observe that any of the end-to-end learning models, even when compared to deep-learning features, are able to capture higher level semantics better than the hand-crafted features. We can also notice that the best results were obtained when combining the two streams into a single model, which resulted in a performance boost in terms of accuracy and mapping metrics.

The obtained results were overall positive and provide a baseline for further improvements. The authors have concluded that the two stream architecture has provided improved results over conventional CNN, as objects can provide valuable information in certain cases, when mapping land use (e.g. bicycles for a bicycle store). It was also possible to conclude that considering fine granularity can be extremely beneficial, even for problems where coarser classes are to be derived. In Table 2.2, we can observe that, for a problem with 5 top-level classes as the objective mapping, employing fine granularity

Table 2.2: Accuracy results based on different granularities. Based in Zhu et al. (2019).

	45-way	16-way	5-way
Fine Granularity	46.7	61.8	75.6
Middle Granularity	-	60.2	68.4
Coarse Granularity	-	-	49.3

resulted in more than a 25% difference in accuracy, when comparing with coarse granularity training. This can be explained due to the fact that some of the concepts can be very different at the low granularity level, while belonging to the same high granularity class. Some possible future improvements are also to be considered for this or other works in the same area, such as the use of humans to provide some guidance to the model, the processing of other types of information besides pictures (e.g. videos), and the use of off the shelf object detectors to boost the object stream.

In Srivastava et al. (2020) another approach was taken in the use of ground-level images for land use mapping. Instead of using community-shared image collections to fine-tune the model, the authors opted for collecting images from Google Street View (GSV). This choice is justified by the fact that it offers better and bigger coverage for urban areas than community-shared photographs, such as the ones available at Flickr, as these tend to be concentrated in turistic areas. Adding to this, GSV pictures also allow to avoid processing photographs that fail to characterize a certain region, such as selfies.

To extract features out of each picture, the authors used a VGG16 (Simonyan and Zisserman, 2014) architecture, pre-trained in the ImageNet dataset, as it provided a strong baseline that would allow to fine-tune the model for the given task.

To obtain the mapping, several street photos had to be analysed and aggregated to provide a landuse class for each area. To accomplish this goal, two alternatives were introduced:

- The extraction of a class for each picture using a simple architecture, as shown in Figure 2.7, and the combination of the results using majority voting or the use of a multi-layer perceptron to obtain a class, after averaging the features.
- The use of a Siamese network (Bromley et al., 1993), that can receive a variable sized input (VIS). This type of network allowed to work on several input vectors (i.e. images) while sharing the same network weights. To obtain a landuse class, the feature representations of each image were then combined using an aggregator. This resulted in the architecture pictured in Figure 2.8.

The model was applied to a multiclass mapping problem with 16 possible classes, in Île-de-France, and an overall accuracy of 62.52% was obtained, using a VIS-CNN with an average aggregator, which valued the most repetitive attributes that were present in the images feature vectors. Due to this outcome, this model was able to achieve better results than previous works in the same area. However, the accuracy of the model was not constant along all the different classes, as shown in Figure 2.9. This is due to the fact that buildings from certain classes (e.g. government, heritage, religious) share

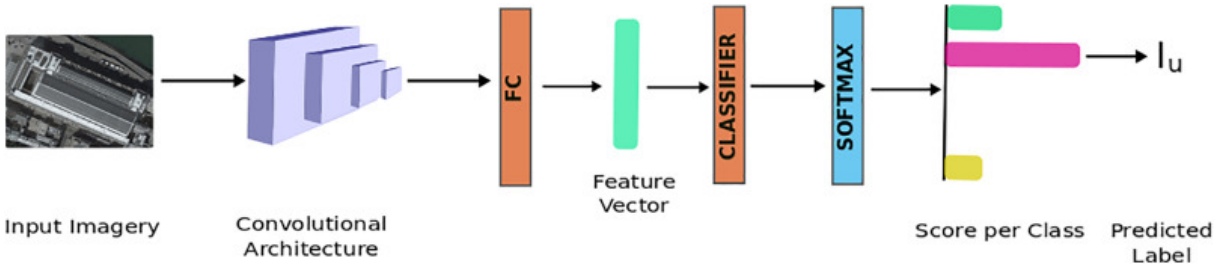

Figure 2.7: Standard procedure when using a CNN model. From Srivastava et al. (2020).

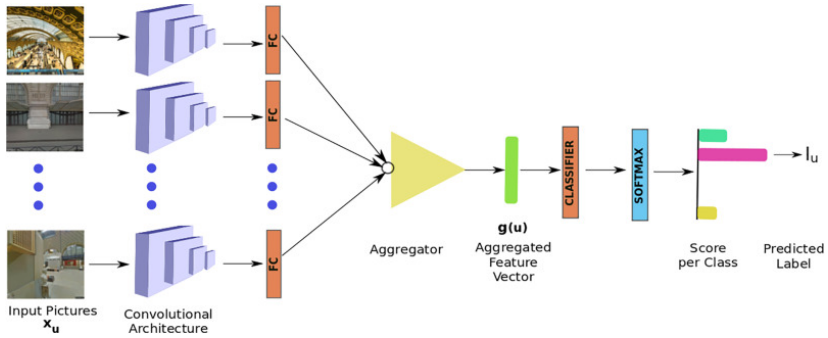

Figure 2.8: Variable Input Siamese Network proposed in Srivastava et al. (2020), in order to extract a single land use class from a variable set of images collection from GSV.

some characteristics (e.g. statues), making it difficult to label some regions by only looking at building facades. The authors proposed that this issue could be tackled by further improving the model using community-shared pictures that could provide details about the building's interior.

2.2.1.3 Further Work on Land-Use and Scenic-Beauty Mapping

An application that will be later discussed in the proposed work is the use of ground-level imagery to map natural beauty. Several works have already addressed this task, some of them using CNNs as a way to obtain better and more consistent results. One of the employed techniques that stands out in Workman et al. (2017), relates to the addition of an input channel containing overhead imagery that is combined with the analysis of ground-level photographs, in order to further improve the obtained results. A similar method was applied in Srivastava et al. (2019), which expanded the work from the same authors (Srivastava et al., 2020). Improved model accuracy was obtained by adding a new processing stream that analysed satellite coverage of the urban-objects, in addition to the stream that processed ground-level images, as seen in Figure 2.10.

2.2.2 Advanced Neural Models for Image Classification

In the last couple of years, several newly introduced neural models have presented highly improved results in a range of tasks. In this section we will discuss the architecture of two of these advanced models, the DenseNet and the EfficientNet.

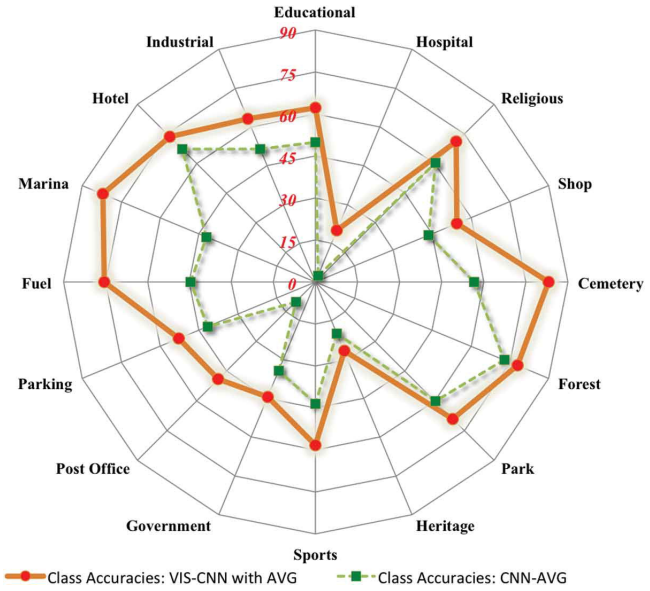

Figure 2.9: Accuracy for each of the 16 classes, by obtaining a classification for each photograph and using the averaging technique (CNN-AVG), and by using the Siamese Network with an Average Aggregator (VIS-CNN with AVG). From Srivastava et al. (2020).

2.2.2.1 DenseNet

The DenseNet model Huang et al. (2016) is a deep neural model which presented greatly improved results in image classification problems, when compared to other existing convolutional networks, due to its unique set of features.

The most distinguishable feature of the DenseNet was its approach to shortcut connections. These type of connections had already been used in several models, such as the ResNet model (He et al., 2015), and enabled the network layers to produce an output combining both the pre-processed and post-processed data, which proved to result in a more stable learning. The main innovation in the DenseNet model, was the fact that each layer was able to receive the output from all the preceding layers, as it is shown in Figure 2.11. This enabled deeper neural models to tackle the loss of information, while allowing a better propagation of the error during the training phase, by avoiding the vanishing gradient problem. Regarding the merging of the outputs from previous layers, unlike previous models where the pre-processed data was added to the output, in the DenseNet model all the outputs are concatenated, resulting in a clean representation of the results from all the preceding layers.

Due to the way the outputs are propagated to subsequent layers, the DenseNet suffers from a rapid increase in the size of the input, and it is only able to combine feature maps with the same spatial size. To deal with these issues, there is usually a small number of output channels, and a CNN is divided into DenseNet blocks that end in a transition layer, responsible for reducing the dimensionality of the output. This architecture is pictured in Figure 2.12.

Finally, it is important to notice that the results of activations from previous layers are left unaltered by subsequent layers, which results in a conservation of the information learned by all layers. Therefore,

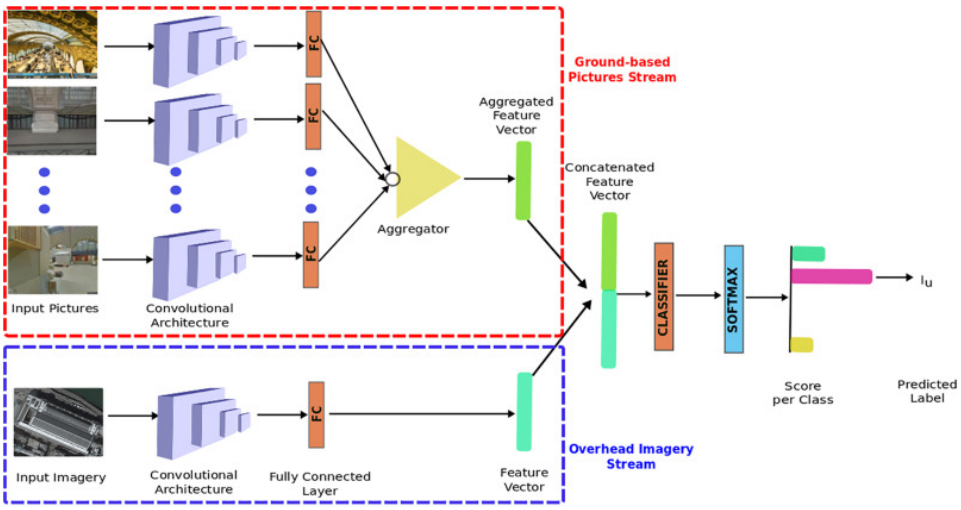

Figure 2.10: A new stream (Overhead Image Stream) was added in order to complement the analysis of ground-level imagery. From Srivastava et al. (2019).

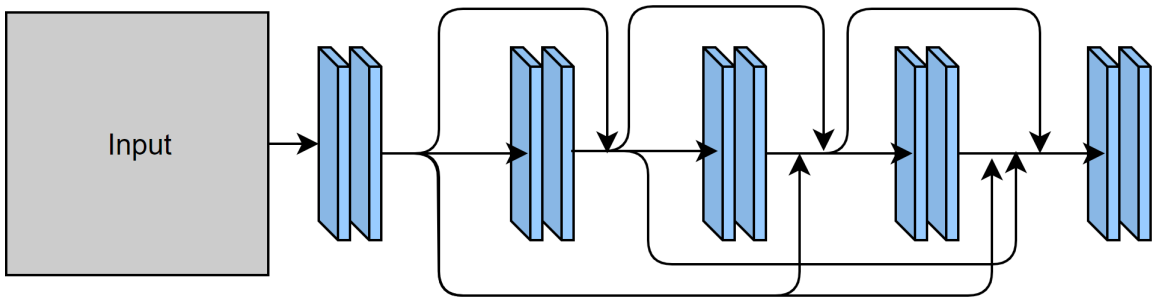

Figure 2.11: DenseNet Block. The arrows represent the flow of feature vector maps between convolution layers.

each layer equally contributes to a global information vector. As a consequence of this behaviour, the number of tunable parameters is also greatly reduced.

At the time it was presented, the DenseNet model was able to outperform all other neural models in a range of classification tasks, obtaining improved results in several datasets, such as in the CIFAR10 dataset, which comprises a collection of 60000 32×32 images, divided in 10 classes. The obtained results are shown in Table 2.3. Furthermore, the DenseNet was able to achieve a smaller error rate while also reducing the number of parameters.

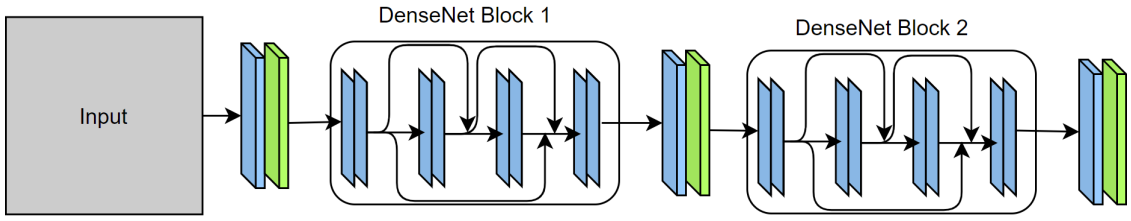

Figure 2.12: CNN with DenseNet blocks. The blocks are interleaved with a convolution (in blue) and a pooling layer (in green) to reduce the dimensionality of the output.

Table 2.3: CIFAR10 results with Data Augmentation

	Test Error
ResNet-110	6.41
DenseNet-100	4.62
ResNet-1001	4.5
DenseNet-250	3.6

2.2.2.2 EfficientNet

After the introduction of deep convolutional neural networks, such as the ResNet or DenseNet models, improving the performance of this kind of models has become a matter of scaling up its architecture. This scaling can usually be done by increasing the depth of the network (i.e. increasing the number of hidden layers), by enlarging each layer (i.e. increasing the number of channels), or by increasing the input resolution. This was commonly achieved by manually tuning one or two of these dimension, which not only was a slow process, but it would also lead to unoptimized results in terms of accuracy and efficiency. Furthermore, as the number of parameters grew, the verified gain would diminish substantially, until a point where the improvements did not have any impact (e.g. ResNet-101 and ResNet-1000 present similar performance).

To tackle the referred issue, a new scaling method, named compound scaling, was proposed in Tan and Le (2019), which later originated a new type of CNNs, called EfficientNets. The idea behind this technique is to uniformly scale the network along its three dimensions with the aim of optimizing the networks performance, while reducing the need for more computational power.

The principle behind this idea is that given a certain value for available memory and available computing power (FLOPS), the objective is to maximize the accuracy obtained by the model by adjusting the network's depth(d), width(w) and the resolution of the input(r). Due to the fact that there is a dependency between all the variables, it becomes difficult to find the optimal solution to this problem, which constitutes the reason why most of the networks used to be extended in only one dimension.

The compound method introduces a new procedure to obtain the optimal values for the variables. To obtain the changes in each dimension, we compute:

$$d = \alpha^{\phi}$$
$$w = \beta^{\phi}$$
$$r = \gamma^{\phi}$$
$$s.t.:$$
$$\alpha.\beta^2 - \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

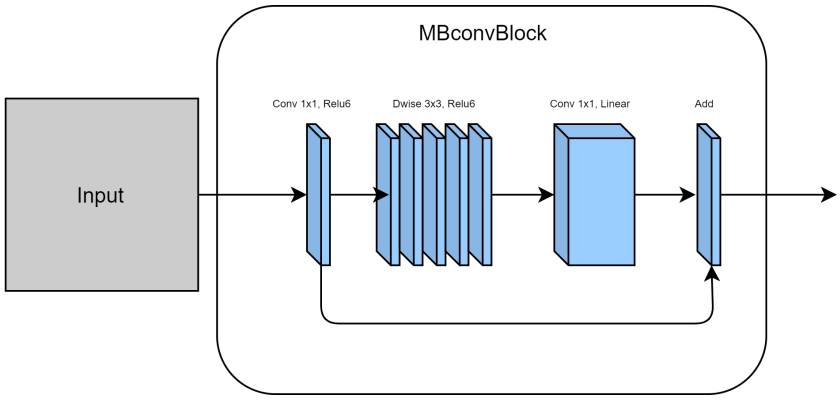

Figure 2.13: MBconv Block. The arrows represent the flow of feature vector maps between layers.

Figure 2.14: Architecture for the baseline EfficientNet-B0 network. Mainly constituted by Mobile Inverted Bottlenecks(MBConv).

Where α, β, γ are fixed hyperparameters and ϕ establishes the constant growth for each dimension.

To begin with, a base network architecture was built to serve as a baseline model. Next, the authors fixed ϕ=1 and performed a grid search for the optimal values of the hyperparameters α, β, γ. Finally, the effectiveness of this method was tested by scaling the network according to ϕ.

To prove the performance difference between the use of this method and previous CNNs architectures, the authors built a baseline model, named EfficientNet-B0 which follows the architecture presented in Figure 2.14. This network is mainly constituted by Mobile Inverted Bottleneck convolutional blocks (MBConvs), presented in Figure 2.13. These type of blocks were introduced in the MobileNetV2 (Sandler et al., 2018) and were created with performance in mind, by reducing the number of parameters needed in each block.

MBConv blocks work around the principle of depthwise separable convolutions. Unlike the normal convolutions, in which we begin by applying each filter directly to all input channels, and then combining the results of all channels, in depthwise separable convolutions we start by applying one single filter to each input channel and only then combine the outputs using a 1×1 filter. This results in a great reduction of parameters, and a reduction in the number of multiplication operations in the order of 8 to 9 times when compared to the number of multiplications executed in standard convolutions.

Looking at a practical example, pictured in Figure 2.15, we consider an input **I** with size $D_I \times D_I \times M$, with D_I representing the height and width of an input feature map, and M representing the number of input channels. A standard convolution would produce a $D_G \times D_G \times N$ output, in which N is the number of filters and D_G represents the height and width of an output feature map. For this convolution, the kernel would have size $D_K \times D_K \times M \times N$. Considering the above values, the resulting computational

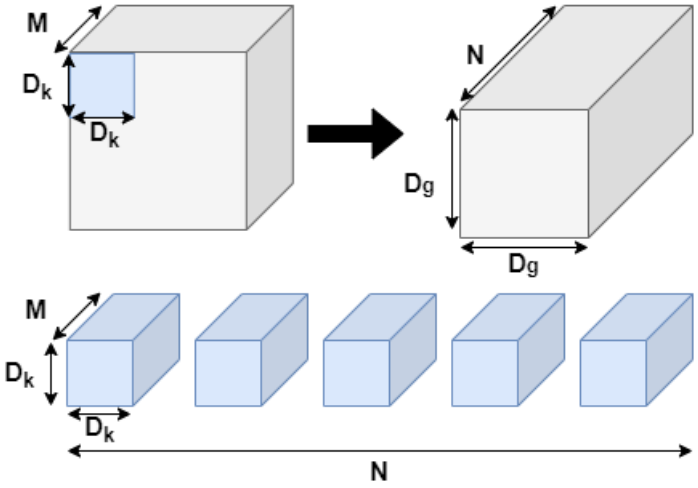

Figure 2.15: Standard Convolution by applying N $D_k \times D_k$ filters to an input with M channels

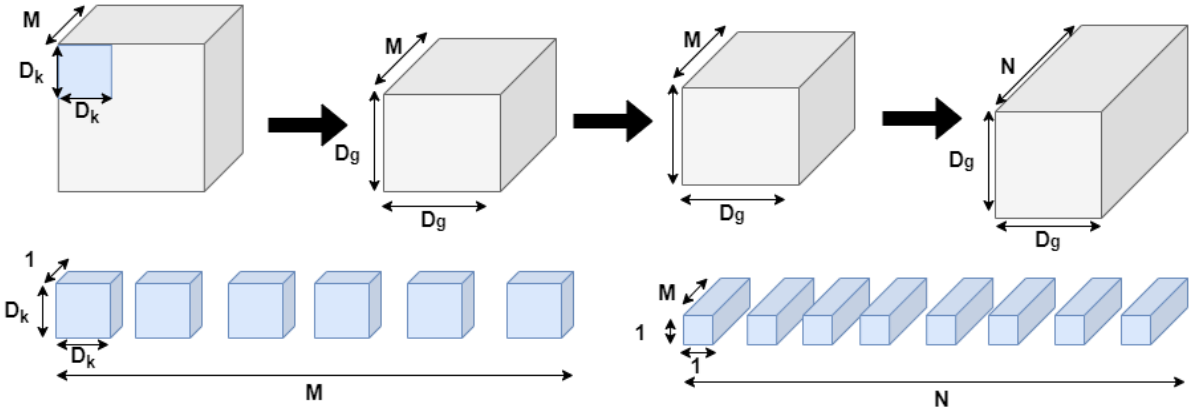

Figure 2.16: Depthwise Separable Convolution. First, one $D_k \times D_k$ filter is applied to each channel M. Second, N 1×1 filters are applied to the result of the previous operation, resulting in a similar output to the standard convolution but with a reduced number of parameters

cost would be:

$$D_K^2 \times D_G^2 \times M \times N \tag{2.9}$$

For a Depthwise Separable convolution, as exemplified in Figure 2.16, we would begin by applying only 1 filter for each input channel, resulting in a kernel with size $D_K \times D_K \times M$, which results in an output of size $D_G \times D_G \times M$. After this step, we obtain the final output by executing a pointwise convolution using 1×1 filters and combining the results across all input channels. Looking at the computational cost of such operations, we obtain the following expression.

$$D_K^2 \times D_G^2 \times M + D_G^2 \times M \times N \tag{2.10}$$

Comparing the computational cost between standard and Depthwise Separable convolutions:

Table 2.4: Comparison between the EfficientNet architectures and current deep neural models, for different levels of accuracy.

	Accuracy Top-1	Accuracy Top-5	Parameters
EfficientNet-B0	77.3%	93.5%	5.3M
ResNet-50	76.0%	93.0%	26M
DenseNet-169	76.2%	93.2%	14M
EfficientNet-B1	79.2%	94.5%	7.8M
ResNet-152	77.8%	93.8%	60M
DenseNet-264	77.9%	93.9%	34M
EfficientNet-B7	84.4%	97.1%	66M
GPipe	84.3%	97.0%	557M

$$\frac{D_K^2 \times D_G^2 \times M \times N}{D_K^2 \times D_G^2 \times M + D_G^2 \times M \times N} = \frac{1}{N} + \frac{1}{D_K^2} \tag{2.11}$$

Looking at the previous equation, and replacing $N = 2048$ and $D_K = 3$ we conclude that Depthwise Separable Convolutions costs 11.1% of the computational cost of a standard convolution (i.e. 9 times faster).

To further increase the performance of these blocks we make use of both inverted residuals and linear bottlenecks, which are based on the premises that a low dimensional subspace can capture the information from feature maps, and that non-linear activations, such as the ReLU, lead to information loss.

In order to improve the gradient flow to the initial layers, blocks use residual connections, such as the ones found in the DenseNet. However, unlike previous networks, where the residual connections are used between layers with a high number of channels, these connections are made between narrow layers, which results in a decrease in memory and computational cost. This technique works based on the above mentioned premises that the valuable information is still contained in the bottlenecks, allowing us to connect these type of layers.

Taking into account the loss of information that results from the use of non-linear activations, and the fact that blocks use residual connections between narrow layers, it is expected that these blocks suffer from a considerable reduction in accuracy. To tackle this issue the authors proposed the use of a convolution with a linear activation before merging the output with the initial activations. This addresses the loss of information generated by non-linear function, without the need to increase the number of channels in the narrow layers that contain the residual connections.

Looking back at the EfficientNet-B0 architecture, the model was then scaled up several times, using the compound method, and a comparison was established between the EfficientNet family and the current alternative models for different values of accuracy in the ImageNet Challenge. By looking at Table 2.4 we can see that EfficientNets were able to obtain state of the art performance results while using a much smaller number of tunable parameters.

Table 2.5: Manual and compound scaling on ResNet-50

	Accuracy Top-1	FLOPS
ResNet-50(baseline)	76%	4.1B
ResNet-50(tuned depth)	78.1%	16.2B
ResNet-50(tuned width)	77.7%	14.7B
ResNet-50(tuned resolution)	77.5%	16.4B
ResNet-50(compound scalling)	78.8%	16.7B

The same compound scaling methodology was also applied to existent networks, such as the ResNet model, and, as can be seen in Table 2.5, by fixing a certain constraint in computing power, this technique was able to improve the result obtained via manual tuning.

To further evaluate the impact of this method, EfficientNets were also applied to a variety of datasets, such as the CIFAR-10 and CIFAR-100, to assess its capability to transfer learning. EfficientNets obtained state of the art performance for every dataset, surpassing all of the current CNNs models in most of them, once again with a much lower number of parameters.

Considering that, by scaling EfficientNets, we are capable of further improve the overall accuracy of neural models at a much lower cost than tuning a CNN manually, and due to the complexity of the task in hands, the use of this family of networks, along with its compound scaling method can be extremely beneficial for this work.

3 Approach to Terrain Mapping Based on Deep Neural Networks

This chapter presents an overview of the proposed approach. Section 3.1 details the procedure used for the extraction of land-use and scenic beauty mappings. Section 3.2, provides a more detailed view of the model and the methods employed in order to try and improve the overall performance in the selected tasks. Section 3.3 presents and describes the sources of data used for this project. Finally, Section 3.4 presents an overview of the contents of this chapter.

3.1 The Proposed Approach to Land-Use Mapping

Only recently have ground-level photographs been used in order to provide an alternative automated solution for mapping tasks that were usually achieved adopting methods based in manual annotations. Works such as Zhu et al. (2019), have established a strong base on how leveraging available ground-level photos could impact tasks such as land-use mapping. Srivastava et al. (2019) works expanded this idea and enhanced the current methods by combining multiple pictures in order to further improve the accuracy of mappings. However, none of these previous approaches has explored the possibility of a model that could look at a set of images as a sequence that represents a region, instead of only analysing images individually.

My proposal consists in organizing photographs into sequences, based on the distance to a certain interest area, and make use of a Recurrent Neural Network, together with a Convolutional Neural Network model in order to better leverage possible links between pictures that might help further improve the accuracy of the mappings for that area. To apply and test this approach, the following step-by-step procedure was considered:

- Create a map for a study region based on available land-use data, provided via hand-annotations. Aggregation of the available information into a meaningful set of land-use classes that can be used in order to train and test the envisioned model.
- Transform the created map into a raster based on a defined resolution, in this case I considered areas which are constituted by 25x25 meters cells. When a region with conflicts between different mapping classes is found, the area is mapped according to the majority class. Due to the complexity of land-use mapping, some classes are aggregated in order to create new and more meaningful

super-classes. One simple example is the aggregation of classes representing different urban fabric densities into a urban fabric super-class.

- Collect ground-level photographs which provide geospatial information about the place they were taken. Select a set of photographs which were taken inside the study region and overlay their locations with the raster created in the previous step, in order to attribute a class to each picture, according to the area it was taken in.

- Split the study region into four sub-regions. The photographs taken in three out of the four sub-regions can be used for training while the forth sub-region can be used as a validation dataset. We can use a cross-validation method to verify the model's performance in each sub-region.

- Train a convolutional network in a simple classification task, in which the target classes are the ones extracted in the previous step. This step is introduced due to hardware constraints in terms of memory usage, which does not allow us to train an end-to-end deep neural network on a full sequence of pictures. As such, this step will work as a pre-training phase for a segment of the model's weights, which will then be locked to allow the training of the remaining weights within the memory constraints.

- For each of the 25x25m cells, produce a sequence of photographs in descending order according to the distance to the centre of the square. For this ordering and based on the available coordinates for each picture, we can make use of a metric such as the haversine distance to calculate which photographs are closer to the centre of each square and can better depict its content:

$$D(x, y) = 2 \arcsin \sqrt{\sin^2(\frac{x_1 - y_1}{2}) + \cos x_1 \cos y_1 \sin^2(\frac{x_2 - y_2}{2})} \tag{3.1}$$

  In the previous equation, x and y are the points between which we calculate the distance and x_1/x_2 and y_1/y_2 are the latitude and longitude of each of those points, respectively.

- Train a recurrent neural network model in a classification task in which the input are the sequences of images for three of the four sub-regions and the target classes are the land use classes attributed to each square based on the data from the raster. Part of the model is initiated with the weights obtained in the image to image pre-training phase. These weights are locked in order to reduce memory usage and allow the tuning of the remaining weights by resorting to bigger sequences.

- Use the sub-region left out during training to evaluate the model's performance and create a new raster for this area based on the extracted land-use classes, in order to create a visual representation of the results that is easily comparable to the original raster.

Based on the data and procedure presented above, the previous task can be expanded in order to contemplate the use of the same region for scenic beauty mapping by considering the following steps:

- Collect scenic-beauty scores available for photographs taken inside the region considered for the previous task. For each image, a set of at least 3 scores that range from 0 to 10 are averaged in order to obtain a single scenic beauty score for each image.
- Utilize the available information to train the same convolutional neural network architecture used for land-use mapping. However, this time the classification layer is replaced by a linear regression, as we now have a real value as target, instead of classes.
- Overlay the land-use raster with the pictures used in the scenic beauty task. Attribute scenicness scores to the 25×25m squares based on the averaging of the scores of the pictures inside its area.
- Train a recurrent neural model in a regression problem with the scenicness scores as target and using the sequences of images that were computed in the previous task.
- Extrapolate the results for the remaining tiles which do not possess any scenic beauty scores in its area and create a raster that presents the obtained sceniccess scores for the study region.

The entire procedure was implemented in the Python[1] programming language, as there already many packages that facilitate the collection of data and the implementation of the desired model. Naming some of the most important tools, this work has made extensive use of the GDAL[2] package, which allowed me to translate and edit vector and raster geospatial data formats, and the Tensorflow [3] and Keras [4] packages, which facilitated the creation and training of deep neural models based on the proposed approach. Some of the more recent models and methods were also tested by recurring to specific packages, such as an EfficientNet Tensorflow implemention[5] based on the original paper (Tan and Le, 2019).

3.2 Model Architecture

The envisioned deep neural model can be divided in two segments, the feature extraction segment (CNN) and the mapping extraction segment (RNN). Figure 3.1 illustrates this model.

In the first segment, a Convolutional Neural Network (CNN) is used analyse each picture individually. For this purpose, I have made use of the EfficientNet model, as it has proven to provide state-of-the-art performance with a small number of parameters on well known classification problems, such as in the ImageNet Dataset (Russakovsky et al., 2015), when compared to similarly performing architectures. This architecture also shown strong transfer of learning between similar classification problems. As such, in order to speed-up the training phase of the envisioned model, I will make use of network weights obtained through a pre-trained model on the ImageNet Dataset.

[1] http://www.python.org/
[2] http://gdal.org/
[3] http://www.tensorflow.org/
[4] http://keras.io/
[5] github.com/qubvel/efficientnet

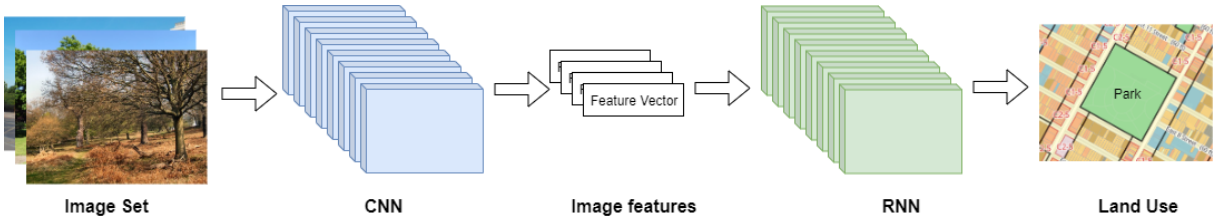

Figure 3.1: Illustration of the model and procedure that is planned to be used to process ground-level imagery and extract a land usage mapping

Besides the basic CNN architectures I also evaluate the performance of the following methods to help in model training: the auto-augment (Cubuk et al., 2019) augmentation technique; the AugMix (Hendrycks et al., 2019) augmentation technique; the supervised contrastive learning method (Khosla et al., 2020).

The auto-augment method consists in a data augmentation technique which aims to automatically find the optimal augmentation policies for a certain dataset. The result of this method consists in a set of sub-policies which are randomly applied to data during the training phase. The authors of this paper applied this method to several datasets, including the ImageNet dataset, and were able to improve the state-of-the-art accuracy and the transfer learning capability amongst different datasets. Taking this into consideration, I have made use of the list of sub-policies applied by this method in the ImageNet dataset, which include a mixture of the following augmentations: automatic addition of contrast; histogram equalization; posterization; solarization; shearing; translation; sharpness change; brightness change; color changes; image inversion.

The AugMix data augmentation method combines ideas from several previous approaches to increase the robustness of a network, allowing it to better deal with unforeseen corruptions in the testing data. To achieve this, new training samples are formed by performing several augmentations of a raw image, which are made by applying a chain of transformations to it, such as a translation or rotation, and then mixing them together, along with the original image, using elementwise convex combinations, as presented in Figure 3.2. Furthermore, the authors of this method propose the use of a Jensen-Shannon Divergence consistency loss during training, in combination with the original supervised training loss, to help the network obtain similar results for the augmented and original samples. For this work, this technique was applied by considering the following augmentations: automatic addition of contrast; histogram equalization; posterization; solarization; shearing; translation.

The contrastive learning technique consists in a two step training approach. In the first step, the top classification layer of the CNN model is removed, leaving the set of features as output of the model. The model is then trained with the intent that pictures from the same class are close in the feature space and that pictures from different classes are as far away as possible. For this purpose, a max margin contrastive loss (Hadsell et al., 2006) is used. This loss function essentially equates the euclidean distance between feature vectors, in order to generate clusters in the feature space that should aggregate the photographs from each class. The second step consists in adding the classification layer back and make use of the pre-trained weights to extract classification results based on the aggregated feature

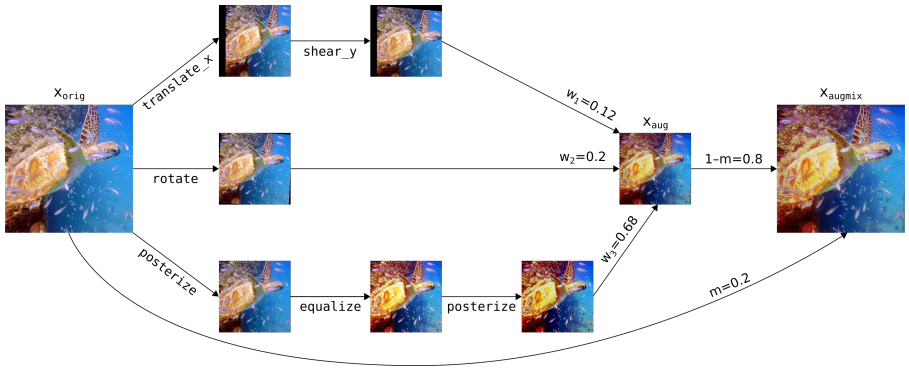

Figure 3.2: AugMix procedure. The raw image begins by being processed in several transformation sequences. The resulting augmented images are then combined, along with the original input, in the last step. From Hendrycks et al. (2019).

sets.

In the second segment, the main objective is to make use of a Recurrent Neural Network (RNN) architecture to combine the features extracted in the previous steps in order to find links between images that can prove to be advantageous for the generation of mappings, when compared to the previous alternatives of mapping via an image-by-image analysis, as in Zhu et al. (2019), or via an aggregator that combines features of multiple pictures by using pooling, similar to what was proposed in Srivastava et al. (2019). To train this part of the model, I make use of a CNN with locked and tuned weights based on the results of the best performing architecture and ancillary method from the training of the first segment and, after removing the classification layer, I feed the last feature output into an RNN LSTM layer. Finally, either a classification layer or linear regression layer is added on top, based on the desired output of the task in hand.

3.3 Data Sources

To train the RNN CNN model for the identification of land use classes, the first step consisted of collecting a large number of photographs, in order to provide a vast ground coverage for the city of London. To achieve this, images were obtained by resorting to the *Geograph.uk dataset*.

The *Geograph.uk dataset* contains a collection of images that offer ground coverage for most of the United Kingdom. Unlike other publicly available datasets, such as the *Flickr Creative Commons dataset*, in which the photographs do not present any pattern or purpose, the Geograph.uk initiative was created with the intent of dividing the U.K. in small $100m \times 100m$ square tiles and collecting at least one photograph that could identify the characteristics of each one of them. As a consequence, this dataset presents a low number of noisy pictures, such as selfies or food photographs, as its images mostly focus on relevant characteristics of the area around the place the photo making it ideal to train the model and help identifying the appropriate mapping for the terrain.

After obtaining data samples, the second step was to obtain a ground-truth map for the study region

in which we could extract a set of land use classes that could be paired with the photographs. For this purpose, I have collected the relevant data for the city of London via the Urban Atlas project from the Copernicus Initiative[6].

The Urban Atlas 2012 provides land use and land coverage mappings for a total of 785 Functional Urban Area in Europe. The map provides a mapping according to a set of 27 classes, 17 urban classes and 10 Rural Classes with minimum mapping units of 0.25 and 1ha, respectively.

Access to Urban Atlas data is provided via a set of files in a vectorized format. The vectorized data is accompanied by a legend with information about several terrain characteristics, including land-use data. The land-use data was embedded in each polygon and the vectorized data was then transformed in a raster, as explained in the procedure in Section 3.1.

This work will focus on identifying land use classes for the urban areas of London and, to facilitate the task, the number of classes is reduced to the following set of 8 super-classes, according to the hierarchy established in the Urban Atlas 2012:

- Urban Fabric
- Industrial and commercial activities
- Land without use, construction sites and mining facilities
- Green Spaces and Sports facilities
- Arable land
- Pastures
- Water
- Forest

As a secondary objective, the models and photographic data from the previous task were also used for the scenic beauty mapping. To obtain ground-truth data for this task, I have made use of scenicness scores for photographs belonging to the Geograph.uk dataset, collected via the Scenic-Or-Not[7] game. This game consists of an online challenge in which people are presented an image from a certain area and are asked to quantify the scenic beauty of the photographic set, according to a scale from 1 to 10, as seen in Figure 3.3. Each score is then collected and each set of atleast of 3 collected score is then averaged to obtain a final scenic score for each image. This is done with the objective of reducing the subjectiveness of the score due to different people having different concepts of scenicness, though the small number of reviews per picture still present large variance. The data compiled up to February 2015 is openly available via a TSV (tab-separated values) file which provides, for each image, its geospatial

[6] http://land.copernicus.eu/
[7] http://scenicornot.datasciencelab.co.uk/faq

Figure 3.3: Scenic-Or-Not game interface. The user should use a scale from 1 to 10 to rate the scenic beauty of the presented photograph.

coordinates of the image, the Geograph Image ID and the attributed set of scores, along with its average and variance.

3.4 Overview

In this chapter, I presented the procedure considered for the elaboration of this dissertation. In Section 3.1 I briefly present the general objectives of this work along with an overview of the work process, detailing the steps that were taken for the evaluation of each of the two selected tasks. In Section 3.2 I give an insight of the envisioned model, as well as some methods that are implemented in this work. Finally, Section 3.3 presented the sources of information which were used to collect both the images and ground-truth data to train the model.

4 Experimental Evaluation

This chapter presents the the experimental evaluation for the proposed mapping approach based on the results obtained for a study region which comprises the urban area and suburbs of the city of London. The procedure is mainly tested in the task of land-use mapping, with the secondary goal of verifying the mapping process for the task of scenic beauty. The applied methodology is presented along with the evaluation metrics for both tasks. After this introduction, the results for the image processing segment of the model are presented. Finally, the results for the image sequence processing are presented for both task, as well as rasters that portrait the region according to a set of land-use classes and scenicness score. For the land-use task, the results are supported by graphs that identify the overall accuracy and class-by-class accuracy and tables with other evaluation metrics, such as the F1 score, while for the scenic-beauty task, tables with error metrics are used to evaluate the model's performance.

4.1 Methodology and Evaluation Metrics

The task of land-use mapping presented in the work consists in a simple classification problem where the goal is to accurately extract a class based on the information extracted from a single image or set of images. For this purpose, a way to evaluate the performance of different models consists in the use of metrics such as the accuracy, recall, precision and f1-score.

Considering **C** as the set of all possible classes, in order to obtain the values for accuracy, precision and recall for a class **i** $\in$ **C**, the formulas below can be applied by considering the variables t_p as the number of true positives, which identifies the correctly labeled samples for class **i**, f_p as the number of false positives, which identifies samples that were wrongly classified as belonging to class **i**, and f_n the number of false negatives, which consists in the samples belonging to class **i** that were wrongly labeled as belonging to other classes.

$$Accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{4.1}$$

$$Precision = \frac{t_p}{t_p + f_p} \tag{4.2}$$

$$Recall = \frac{t_p}{t_p + f_n} \tag{4.3}$$

Finally the F1 score allows us to balance and combine the above precision and recall into a single value, which provides a good metric for the accuracy of the model, regarding a certain class. This metric can be obtained by using the result from the previous metrics and the formula below.

$$F1\,Score = 2 \cdot \frac{Precision \times Recall}{Precision + Recall} \tag{4.4}$$

For the task of scenic-beauty, instead of mapping a class, the objective is to extract a real value which represents the scenicness that can be observed in a certain picture. A possible way to evaluate the results is to estimate the difference between the obtained predictions and the original value. For this purpose, we can use various statistics, such as the Root Mean Square Error (RMSE) between the predicted and original values, or the Mean Absolute Error (MAE), as presented in the following formulas:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{4.5}$$

$$MAE = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n} \tag{4.6}$$

In the previous formulas, $\hat{y}_i$ corresponds to a predicted value, y_i corresponds to a original value, while n represents the total number of predictions.

To provide a more accurate evaluation of the tested models, I have divided the study region into 4 sub-regions to follow a k-fold cross validation approach, which consists in feeding 1 of the sub-regions as the testing set and using the remaining as training set. As a result, the model behaviour can then be better estimated by averaging the evaluation metrics obtained for each of the folds. A map of the sub-regions, colored according to the classes for land-use classification is presented in Figure 4.1. For the task of scenic beauty, due to reduced number of pictures in the study region, I have opted to evaluate the models by considering, for testing purposes, all the photos within the dataset used for the previous task and, for training, the remaining photographs of the UK territory for which I was able to collect a scenincess score.

4.2 Convolutional Neural Networks for Image Analysis

This section presents the results obtained for both Land-use and Scenic Beauty tasks by using deep learning CNNs for image analysis. As mentioned in the previous chapter, due to memory usage constraints, this work has been divided in two segments in order to improve the end results of the experiments. Beginning by experimenting with a image to image analysis allowed to use more memory intensive architectures and more easily test certain methods, such as data augmentation techniques. In this section I present the results for the following experiments:

- Use of a DenseNet169 architecture with batch size 32 for information extraction.
- Use of an EfficientNet-B0 architecture with batch size 32 for information extraction.
- Use of an EfficientNet-B0 architecture with batch size 32 along with augmentations generated by using the results obtained by applying the AutoAugment method in the ImageNet challenge.
- Use of an EfficientNet-B0 architecture with batch size 16 along with augmentations and a custom loss fuction generated using the AugMix method.
- Use of an EfficientNet-B0 architecture with batch size 32 along with a supervised contrastive learning technique as presented in Khosla et al. (2020). This method consists in a two step approach in which the layers of the network that precede the classification layer are tuned in a first step based on the differences between the samples inside each batch in order to try and maximize the distance between photographs from different classes, and minimize it between the ones that belong to the same class. In the second step, the network weights are then tuned in order to extract the final class based on the differences between the ground-truth and predicted results.

The models were trained with adjusted parameters in order to maximize the batch size, taking into account the memory restrictions of the system used for this work. This decision was based on preliminary results obtained via the comparison of a EfficientNet-B0 (4 million parameters) with a batch of 32 images with a EfficientNet-B4 (17 million parameters) with a batch size of 4, which pointed to improved results with the EfficieNet-B0. As such, due to the fact that the increased batch size allows to increase the speed and stability of the training process, I opted to make to use smaller network architectures with bigger batches.

All of the CNNs used in this work receive as input images with resolution 224×224 pixels. In order to feed the collected photographs to the networks, for each image, we selected the biggest possible area starting from its center, which allowed to keep the width and height equal, and cropped the rest of the image. The resulting squared shaped picture was then resized in order to match the input resolution of the CNNs.

For the task of land-use, the class attributed to each image is based on the class of the map cell as presented in Figure 4.1. It is important to consider that the used photographs are community shared and were not taken in the same year neither by the same person and device. Adding to that, the geolocation always has an associated error that can vary according to how it was obtained. On the contrary, the ground-truth map used for this task, the *Urban Atlas 2012* was created by an official entity and on a small period of time, based on hand annotations collected according to a standard procedure. Weighting all these factors, it is clear that mixing these sources in order to train a model will result in an associated error, as the lack of a standard in the data collected from Geograph.uk will result in noise, and the fact that the photographs were not taken within a small timeframe means that the landscape might have changed completely between the time the photos were taken, and the time Geograph.uk mapped land-use classes for the considered area.

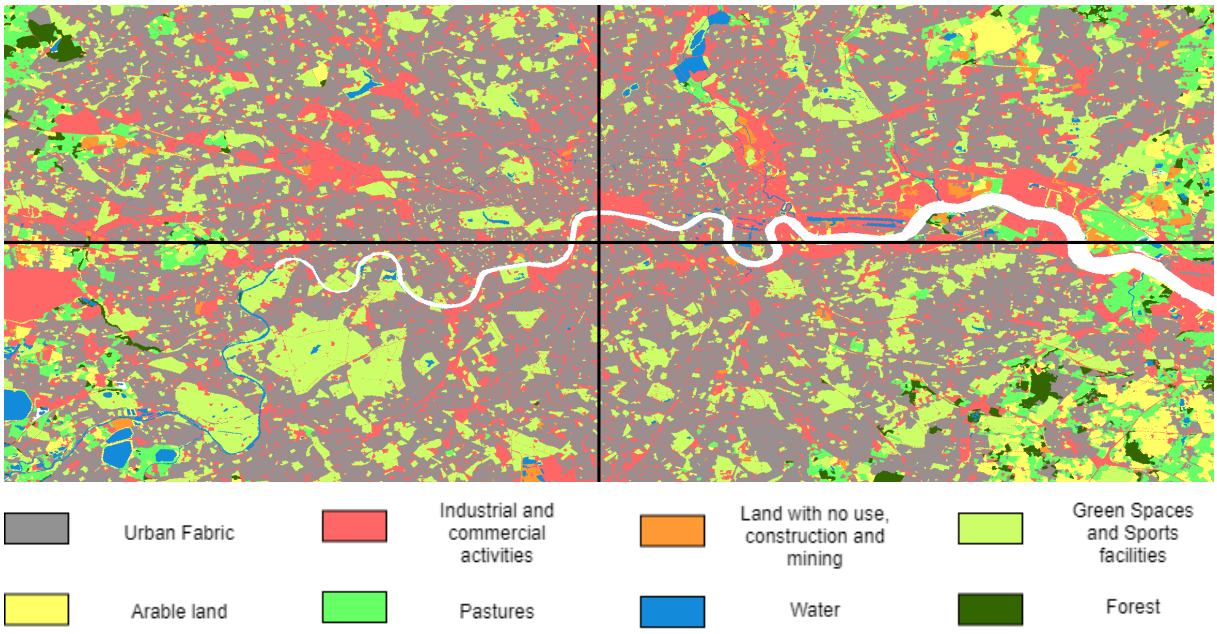

Figure 4.1: Sub-region division for the Land-use and Scenic Beauty tasks, colored according to the Land-Use classes.

	Sub-region 1		Sub-region 2		Sub-region 3		Sub-region 4	
	Train	Test	Train	Test	Train	Test	Train	Test
Urban	93422	44138	130187	7373	116752	20808	133340	4220
Industrial and Commerce	104670	39176	126526	17320	124382	19464	140329	3517
No Use, Construction and Mining	1688	516	2179	25	2027	177	2194	10
Green Spaces and Sports	38007	14727	51413	1321	40507	12227	51713	1021
Arable	11699	380	12079	0	11496	583	12079	0
Pastures	21974	867	22840	1	22081	760	22841	0
Water	7565	2262	9552	275	7512	2315	9810	17
Forest	13283	548	13831	0	13220	611	13831	0

Table 4.1: Support for each land-use classed based on the sub-region used for testing.

Based on the discussed cross-validation approach, for evaluating the results of applying each CNN model to the data, 4 test sub-regions were defined according to the map in Figure 4.1. The training data was composed of the remaining 3 sub-regions, and in order to help and improve the results, some extra photographs were collected from an outer region, as presented in Figure 4.2, and added to the training set. These images were not used in the following image sequence due to the fact that generating and using the extra cells in this region would drastically increase the time required to produce data and train a model. Table 4.1 presents the support (i.e. the number of samples for each class) for each sub-region divided by the 8 classes considered for this task.

The results obtained for the experiments are presented in Table 4.2 and Table 4.3, according to the metrics of accuracy, precision, recall and F1-score, and also according to the results obtained separately for each of the 8 classes.

As shown in Table 4.2, both the DenseNet model and EfficientNet Model performed almost similarly, as expected, based on their similar performance in the ImageNet challenge. The only major difference between these two architectures corresponds to a 34% decrease in the time taken for each training

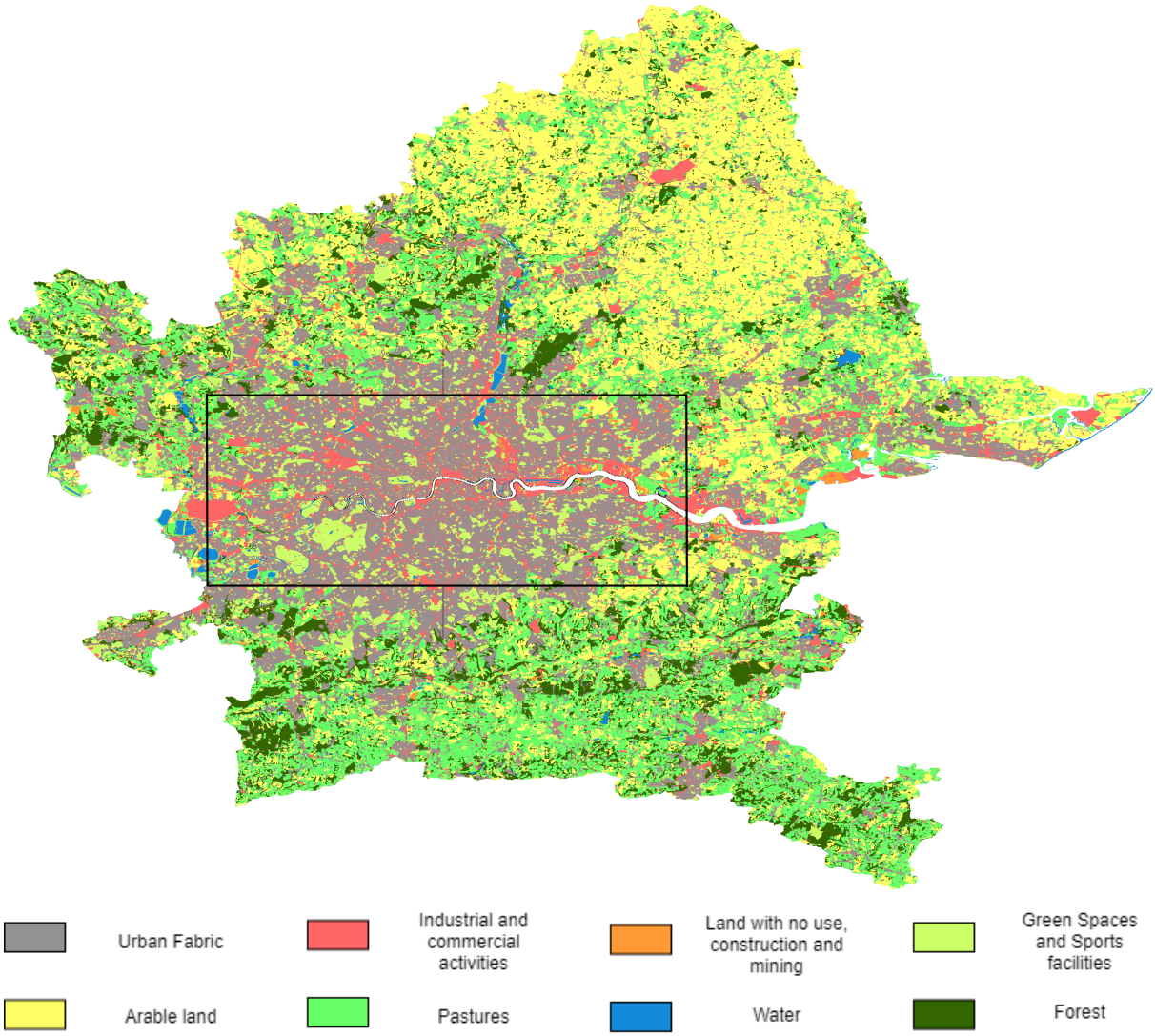

Figure 4.2: Region used for Image to Image analysis, colored according to the Land-Use classes.

epoch for the EfficientNet Architecture.

Regarding the alternative techniques applied along the standard architecture, the AugMix augmentations provided the most increases in performance for some of the classes, which should be correlated with the fact that the original paper reports that this technique tends to allow to deal better with data corruption and improve the transfer of learning between training and test datasets which do not present the same data distribution. For the remaining evaluated methods, there was no noticeable improvements when using the auto-augment augmentations or the contrastive learning pre-training technique. Taking all into consideration, it was decided that the weights learned for the test with the EfficientNet-B0 architecture, along with AugMix augmentations were to be saved and used for the training and mapping phases of the remaining work.

For the task of scenic beauty, due to reduced number of pictures in the study region, I have opted to evaluate the models by considering, for testing purposes, all the photos within the dataset used for the previous task that have scenicness scores, and for training, the remaining photographs of the UK

	DenseNet-169			EfficientNet-B0		
	Precision	Recall	F-score	Precision	Recall	F-score
Urban	0.578	0.680	**0.618**	0.590	0.623	0.604
Industrial and Commerce	0.608	0.535	0.565	0.596	0.588	**0.589**
No Use, Construction and Mining	0.0	0.0	0.0	0.057	0.010	**0.012**
Green Spaces and Sports	0.589	0.512	0.549	0.625	0.493	**0.552**
Arable	0.196	0.029	0.044	0.167	0.064	**0.090**
Pastures	0.209	0.305	0.231	0.191	0.331	**0.241**
Water	0.460	0.308	0.361	0.401	0.357	**0.373**
Forest	0.219	0.261	0.235	0.171	0.365	**0.235**
Macro Average	0.357	0.329	0.325	0.350	0.354	**0.337**
Accuracy		0.57			**0.58**	

Table 4.2: Results for the simple architectures evaluated in the land-use task.

	Auto-Aug			AugMix			Contrastive		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
Urban	0.585	0.644	0.608	0.760	0.841	**0.796**	0.588	0.633	0.608
Industrial and Commerce	0.600	0.559	0.576	0.572	0.636	**0.596**	0.597	0.575	0.579
No use, Construction and Mining	0.028	0.000	0.007	0.135	0.007	**0.007**	0.035	0.007	0.007
Green Spaces and Sports	0.630	0.500	**0.554**	0.638	0.488	0.551	0.630	0.481	0.548
Arable	0.170	0.046	0.066	0.166	0.058	**0.084**	0.148	0.046	0.066
Pastures	0.198	0.353	**0.252**	0.247	0.191	0.217	0.182	0.354	0.236
Water	0.417	0.371	**0.391**	0.441	0.246	0.313	0.424	0.354	0.383
Forest	0.195	0.320	**0.240**	0.186	0.324	0.236	0.181	0.318	0.235
Macro Average	0.353	0.349	0.337	0.393	0.349	**0.350**	0.348	0.346	0.333
Accuracy		0.570			0.57			**0.58**	

Table 4.3: Results for the complementary methods and techniques evaluated in the land-use task.

territory for which I was able to collect scenicness scores. Unlike in the previous task, the data for the scenicess score is extracted directly from the train and test dataset without recurring to a secondary data source for this purpose which helps reducing the error associated with the ground-truth data. However, the scenicness of a photo is not a concrete measure but rather a subjective term that can influenced by a variety of factors, as explored in Seresinhe et al. (2018). By averaging a set of scores attributed by different individuals we can more accurately portrait the scenicness of a photograph, though it will still be a measure heavily influenced by the human expert's perspective of scenic-beauty. As referred before due to the desired output being a real value between 0 and 10, the classification layer of the tested models will be replaced by a linear regression layer. The graph presented in Figure 4.3 presents the support for the scenicness scores. As we can see, the scores follow a normal distribution with a sligth inclination to lower scenicness values, as it was to be expected given that there is a prevalence of urbanized area, which tends to negatively affect this score.

The results obtained for the experiments are presented in Table 4.4 according to the metrics of RMSE and MAE. The models behave in a similar way to the previous task, though by looking at the differences between the basic EfficientNet-B0 and the auto-augment model it is possible to observe that this augmentation technique helps smoothing the predictions and reduce the RSME. This increase in performance is to be expected, as we initialized the CNN with the weights from obtained from the ImageNet challenge and, as the authors of the auto-augment technique report, applying this method usually helps the transfer of learning between different but similar task.

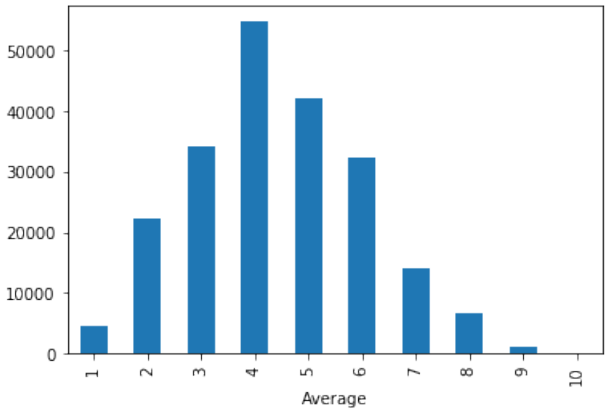

Figure 4.3: Distribution of the scenicness scores obtained from the Scenic-Or-Not Game.

	RMSE	MAE
DenseNet-201	0.90	0.70
EfficientNet-B0	0.89	**0.68**
EfficientNet-B0 with auto-augment	**0.88**	0.69

Table 4.4: Results for the complementary methods and techniques evaluated in the scenic-beauty task.

4.3 Sequence Analysis

Based on the results of the previous tests, the next step of the experiments consists of applying the method that provided the best results in Section 4.2 in order to build a sequence analysis model. For this purpose, we exclude the top layer from the CNN architecture, we load and lock its weights based on the previous training phase results, and use a wrapper (in this case, I make use of the TimeDistributed wrapper[1] class from the *Keras* package) so that it can be applied to multiple images for each learning step. Next, the output features provided by the layers contained within the wrapper need to be processed and combined. This is achieved by using a recurrent neural layer, which analyses the features as a sequence of frames. This layer is composed by LSTM[2] units implemented in the *Keras* package. Finally, the output of the recurrent layer, is fed into either a classification or linear regression according to the task of land-use or scenic beauty mapping, respectively.

Regarding the creation of the photographic sequences for the land-use task, as mentioned in Section 3.1, the raster map obtained from the *Urban Atlas* is divided into 25x25m cells, to which a class based on land-use data is attributed. Next, for each cell, the photos are sorted according to the distance to its center and the sequence of the 10 closest images are attributed to it.

A major issue faced during this work was the fact that, given that the Geograph.uk dataset is generated via voluntary contributions, the photographs are not equally distributed throughout the study region. As such, the quality of the produced sequences of images when it comes to the representing the contents of the cell varies significantly according to the availability of photographs from its surroundings.

[1] `http://keras.io/api/layers/recurrent_layers/time_distributed/`
[2] `http://keras.io/api/layers/recurrent_layers/lstm/`

	Sub-region 1		Sub-region 2		Sub-region 3		Sub-region 4	
	Train	Test	Train	Test	Train	Test	Train	Test
Urban	282865	217803	472882	30800	257606	211346	473934	28843
Industrial and Commerce	109956	89888	191888	9711	101920	88156	193931	7557
No use, Construction and Mining	1824	2231	3990	139	2188	1584	4110	21
Green Spaces and Sports	113452	70876	180874	4735	76399	98704	178880	6731
Arable	7723	4332	12418	0	3920	7224	12418	0
Pastures	20602	21659	42533	0	17943	20134	42533	0
Water	8377	4553	12729	290	4475	7894	12957	62
Forest	14375	8792	12729	290	4475	7894	12957	62

Table 4.5: Support for each land-use classed based on the sub-region used for testing.

Initial tests suggested that there was a large decay in terms of the metrics used to measure the model's performance mainly due to the fact that even the closest image to the cell's center was in an area that did not belong to the same class as the cell it was trying to represent. A possible solution to this could be reducing the granularity of the map, by increasing the cell size and, as a result, increase the chances of having a photograph within each cell. However, this would seriously reduce the support for some of the minority classes, as if we were presented with a bigger cell which, according to the Urban Atlas, could be identified as 20% a minority class and 80% a majority class, the cell would only be seen as belonging to the majority class. As such, the alternative found for this problem consisted in filtering the cells and excluding those for which there is not at least one photograph within 2 kilometers of the cell and inside a region with the same class as the target output. This allows not only to make use of the weight tuning obtained in Section 4.2, but also verify the impact of using sequences of images, when compared to single image analysis. Although this method is applied during the evaluation process, the final rasters and result will also consider all the available cells. In order to speed up the training phase, multiple followed samples of the same photographic sequences (i.e. adjacent cells with the same representative sequences) have been removed and reduced to just 1 sample, as it still allows to train the model.

Based on the previous approach, table 4.5 presents the number of cells for each class. Based on this data and on the sequences of images generated for each cell, I have evaluated the performance of the model based on the following tests:

- Use a set of the 3 closest photographs to the center of each cell as input to a CNN architecture, with its outputs combined via an aggregator, using average pooling. This architecture resembles a simple process of combining features from multiple images, similar to what was applied in Srivastava et al. (2020). The decision of using only 3 images is based on the fact that, due to the low density of photographs in some areas, considering a large number of images would unbalance the features collected from the nearest images.
- Variable sequence of 3, 5 and 10 photographs closest photographs to the center of each cell as input to a shared CNN and its outputs combined via an LSTM layer.

The results for each of the tests are presented in table 4.6. The evaluation process also uses the same cross-validation procedure that was used in Section 4.2 and the results represent the average values obtained for the 4 sub-regions

	Aggregator			RNN 3 photos			RNN 5 photos			RNN 10 photos		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
Urban	0.769	0.888	0.826	0.784	0.877	0.826	0.789	0.884	**0.830**	0.792	0.863	0.827
Industrial and Commerce	0.666	0.489	0.563	0.635	0.519	0.572	0.659	0.513	**0.578**	0.597	0.553	0.573
No use, Construction and Mining	0.000	0.000	0.000	0.008	0.004	0.004	0.072	0.004	**0.018**	0.061	0.008	0.012
Green Spaces and Sports	0.682	0.667	0.669	0.665	0.674	0.674	0.699	0.659	0.677	0.719	0.665	**0.690**
Arable	0.314	0.056	0.096	0.146	0.100	0.120	0.174	0.131	**0.149**	0.178	0.087	0.110
Pastures	0.493	0.495	**0.486**	0.513	0.400	0.446	0.494	0.454	0.471	0.516	0.446	0.481
Water	0.578	0.463	0.513	0.583	0.513	0.544	0.530	0.556	**0.547**	0.549	0.524	0.532
Forest	0.485	0.390	0.429	0.500	0.277	0.353	0.474	0.469	0.456	0.520	0.420	**0.458**
Macro Average	0.498	0.431	0.448	0.479	0.420	0.442	0.486	0.459	**0.466**	0.491	0.446	0.460
Accuracy		0.714			0.713				**0.718**		0.715	

Table 4.6: Land-use task results using photographic sequences

	RMSE	MAE
Aggregator	1.55	1.15
RNN 3 photos	1.13	0.90
RNN 5 photos	1.11	0.88
RNN 10 photos	**1.10**	**0.87**

Table 4.7: Scenic-Beauty task results using photographic sequences

As it can be observed in the presented results, the use of a sequence of images resulted in improved results when compared to the use of a single image. When increasing the size of the sequence, it is possible to notice that the model provides increased accuracy, especially when looking at smaller sequences (between 3 and 5 pictures). However, it is also possible to verify that this improvement associated with the increase in size of the available information starts to fade away when we get to longer sequences (10 images), in which can notice small decreases and increases in the results from different classes, based on the considered metrics. The most likely reason for this behaviour is that due to the fact that we make use of a not so vast dataset of pictures.The quality of the sequences will also suffer with the increase in the number of pictures used for each cell, as the most distant pictures will often be far way from the cell's area and will not provide any meaningful grounds to what should be the class attributed to the image set. Comparing the RNN with the use of average pooling in the model with the aggregator, the improvements in the results obtained by analysing sequences with a RNN are mainly connected to better predictions in the regions with lower density of photographs, while still maintaining the same accuracy for more densely packed regions.

For the task of scenic beauty, there was no map available that could provide scenic scores based on the region, and the ground-truth data had to be collected based on the results obtained for individual photographs. As such, in a similar way to what happened in the previous task, only some cells were selected for the evaluation process, based on the existence of a classified photo within its region. Running the same recurrent neural network models provided the results presented in Table 4.7.

Looking at the results, these present similarities to the ones obtained for the previous task. However, the difference between the RNN architecture and the Aggregator is more prominent than in the previous task. This is most likely related to the fact that the number of samples for this task is lower than the ones used for land-use mapping, which also affects the image density in the studied areas, which will have a bigger effect in the Aggregator, due to the use of a Average Pooling layer to combine the features from different images. Another factor that might benefit the RNN model is the fact that samples that are further away from the interest area might give some better initial clues about the desired output since

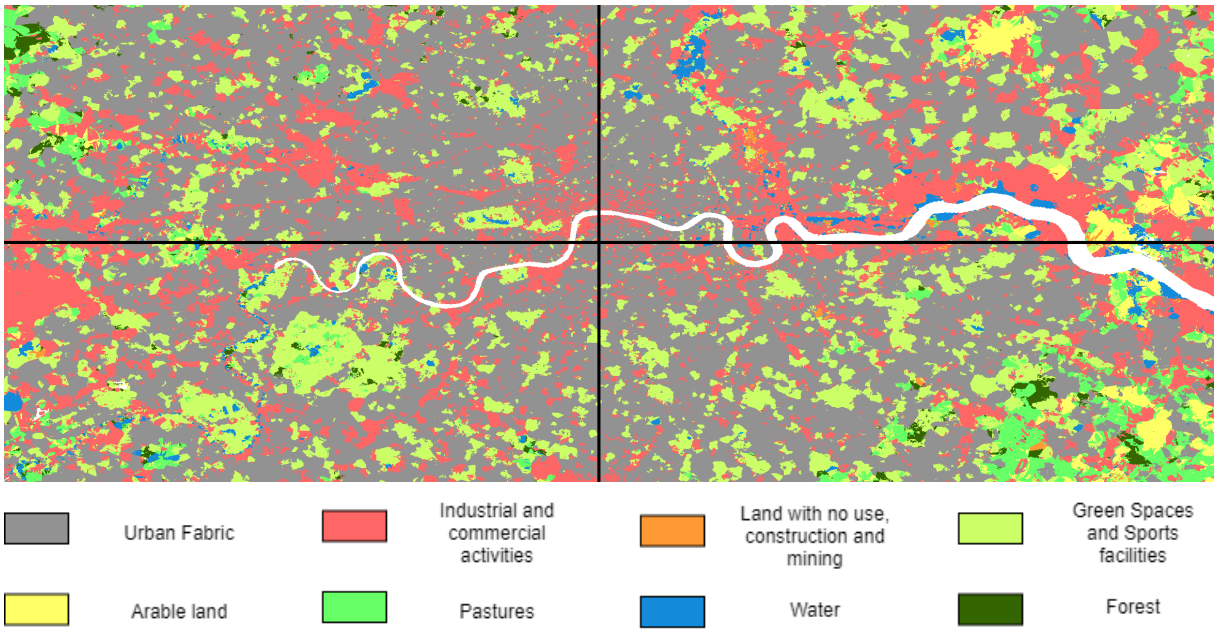

Figure 4.4: Automatically generated Land-use map.

there is usually a smooth transition in scenicness along a region (e.g. city to suburbs and suburbs to coutryside), unlike in the previous task where there might be a fast shift in the land-use classes (e.g. commercial area to urban fabric).

4.4 Maps with Obtained Results

Based on the data collected from the previous experiences, and considering the methods that provided the best results during the evaluation process, we can then generate raster maps based on the predictions obtained with the sequence processing model. In Figure 4.4 I present the results of the automated land-use mapping for the 4 sub-regions. In Figure 4.5 the 2 sub-regions that show the best photographic density and class distribution are shown alongside the ground-truth map for the same area. We can see that the model is able to produce an acceptable mapping of the area, very similar to the ground-truth map, though it still shows some difficulties in differentiating small areas belonging to minority classes.

For the task of scenic beauty, we can make use of the trained model, along with the sequences generated for the land-use task and produce a scenic beauty map, as presented in Figure 4.6. It is important to ackowledge the limitations in terms of accuracy associated with these predictions, due to the subjectiveness of the target value and due to the lack of ground-truth data. However, by comparing the extracted scenic mapping with the original land-use, as shown in Figure 4.7, an interesting observable detail is that regions that are classified as belonging to classes that are expected to provide a more scenic environment, such as forests or green urban areas, are colored as regions with a higher degree of scenicness, which suggests that the extracted information allowed the production of a meaningfull

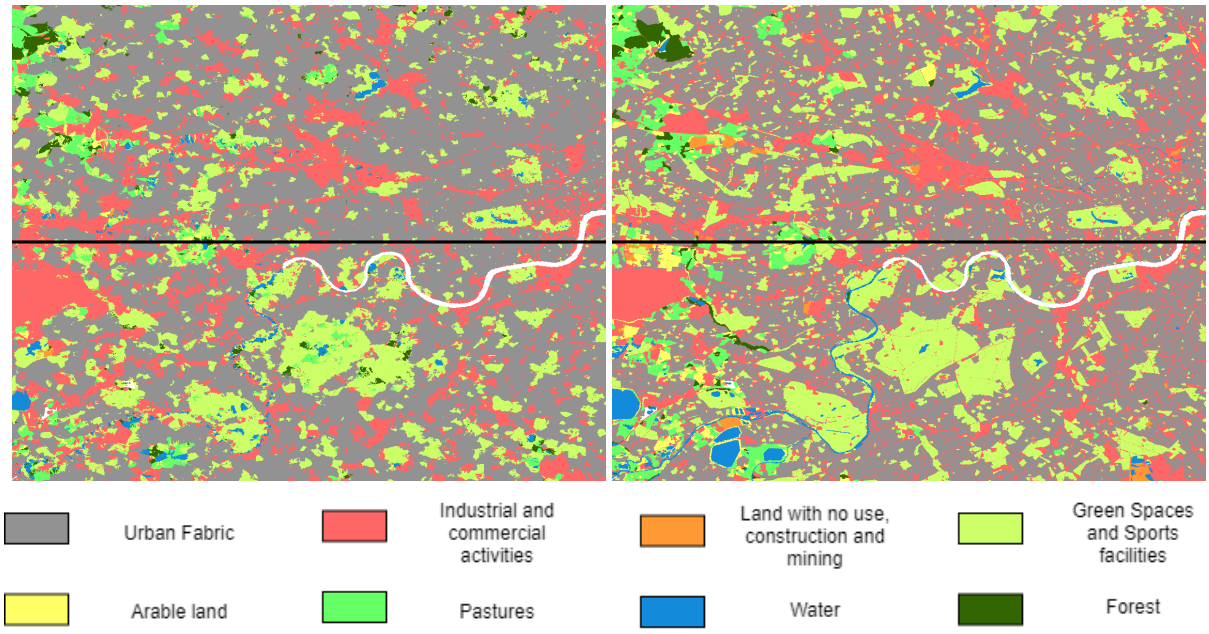

Figure 4.5: On the left, part of the raster automatically generated based on the obtained predictions for land-use, using the sequence analysis approach. On the right, part of the raster created based on the land-use data obtained via the Urban Atlas 2012 data.

scenic beauty map, based on this relation between scenicness and the ground-truth land use classes.

4.5 Overview

In this chapter, I presented the evaluation process that was conducted during the elaboration of this dissertion, as well as the discussion of the obtained results. I have compared different approaches and provided motives for the behaviours shown when using different methods, along with possible changes that might lead to improvements on the results. I have assessed the data collected and tested the proposed approach based on the results obtained in Section 4.2 and Section 4.3 with the objective of producing automated versions of a land-use and scenic beauty map for the city of London, as presented in Section 4.4.

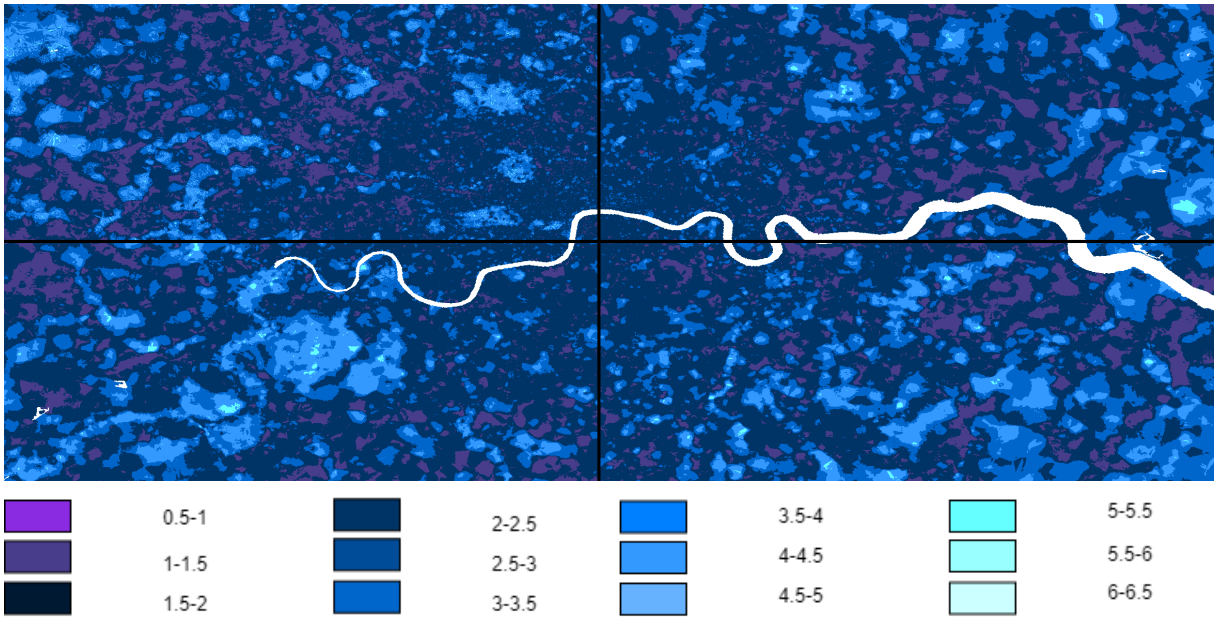

Figure 4.6: Automatically generated scenicness map.

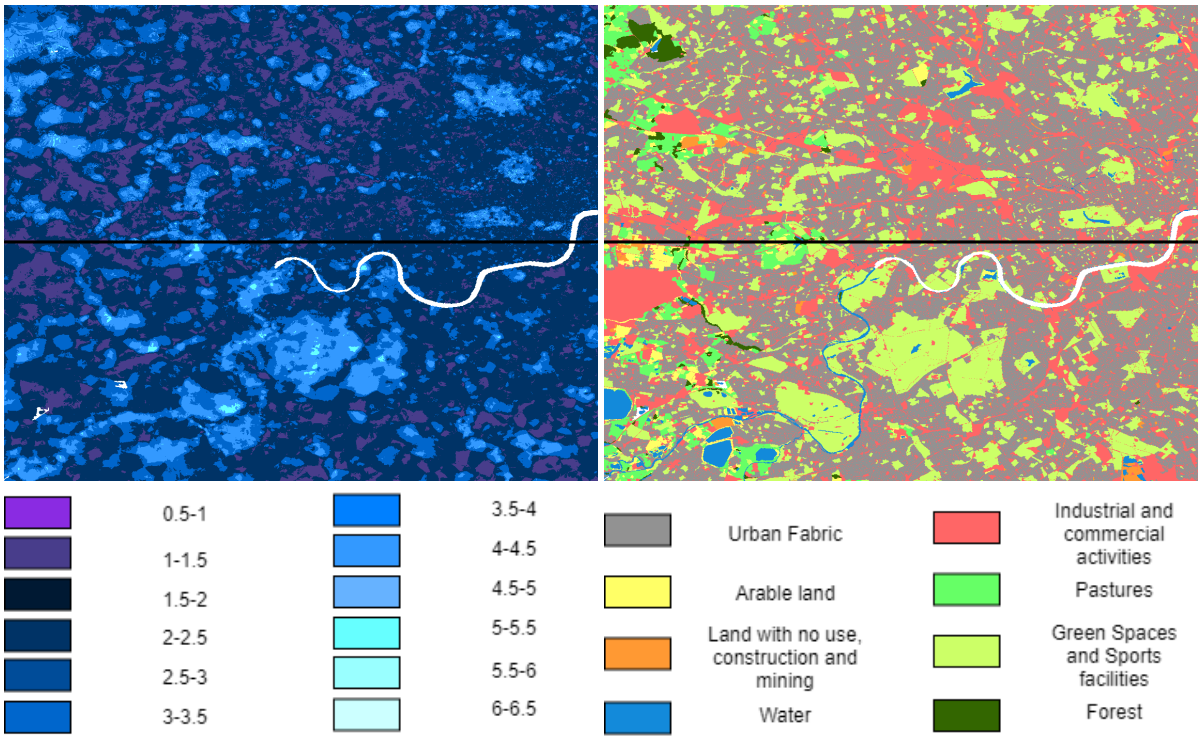

Figure 4.7: On the left, part of the raster automatically generated based on the obtained predictions for scenic beauty, using the sequence analysis approach. On the right, part of the raster created based on the land-use data obtained via the Urban Atlas 2012 data. It is possible to observe a correlation between the scenic beauty and the use given to a place.

5 Conclusions and Future Work

5.1 Summary of Contributions

In this work, I have presented the results of a new mechanism for terrain mapping based on the analysis of sequences of images, using convolutional neural networks, recurrent neural networks, ground-level photos obtained via the Geograph initiative and land-use data extracted from the Urban Atlas 2012 mappings.

The envisioned procedure was split into two segments which were tested in the urban area of the city of London. For each segment, several alternatives were evaluated in order to verify which would be the best combination of techniques that could be applied in order to improve the results. Based on the tests evaluated in Section 4.2, the chosen CNN model for image analysis was the EfficientNet-B0 model, along with the use of a set of optimal augmentations extracted for the ImageNet Dataset. Based on the evaluation procedure described in Section 4.3, the envisioned model has provided slightly improved results, when compared to the use of an aggregator (0.4% in Accuracy and 4% in terms of F1-score) . Adding to that, the use of bigger sequences of photographs for the mapping task has shown to be beneficial when considering the proposed approach. Finally, I have made use of the model to create rasters with a minimum mapping unit of $25m \times 25m$, which portrait the study region according to 8 land-use classes (Urban Fabric; Industrial and commercial activities; Land without use, construction sites and mining facilities; Green Spaces and Sports facilities; Arable land; Pastures; Water; Forest).

The secondary task of scenic-beauty mapping was also explored in this work by leveraging both the dataset and evaluation results collected for land-use mapping. By applying the same principles I have produced a scenic-beauty map for the same study region by training a model based on the scenicness score obtained in the Scenic-Or-Not game, and expanding the obtained knowledge to map the remaining region based on the photographic sequences generated in the previous task.

5.2 Future Work

Regarding future work, a possible way of expanding the presented method would be trying to increase the used set of images, in order to obtain bigger and more accurate sequences that could help improve the represention of each mapping tile. This could be done by recurring to Google Street View

API[1] as explored in Srivastava et al. (2019).

Another alternative for future work would be using other augmentation methods, such as the one presented in Yang and Soatto (2020), which consists of a Fourier Domain Adaptation method. This method allows the augmentation of images, using the inverse Fourier Transform, from both real and synthetic datasets so that there can be knowledge transfer between them. This could be used along with the introduction of new synthetic and real data sources to further improve the model's robustness when it comes to the minority classes.

Based on the issue of the density of photographs for each area and the differences in the results that were verified when processing sequences of images, a possible alternative for sequence processing that could lead to improvements would be to replace the cell by cell analysis with the combination of multiple sequences of pictures from neighbouring cells. This way, the mapping for each cell could weigh in the results obtained for its neighbours, which would reduce the chance of errors associated with incorrect predictions for a single cells' images.

In terms of improving the results specifically for the scenic-beauty task, this work could be expanded by following the procedure used in Qui et al. (2019). These authors suggest collecting more data by generating hand annotations tasks along with interpolation techniques to generate a dataset that could offer a wider terrain coverage.

Finally, a possible idea that would be interesting to explore is the use digital elevation maps (DEMs), such as the *EU-DEM*[2], in order calculate the viewshed of each photograph. The use of these maps along the geolocation tags allows us to understand for which cells the line of sight might be blocked. Calculating this measure can be accomplished by using algorithms such as the one presented in Floriani and Magillo (2003) or, specifically for this task which makes use of ground-level photographs, by using a more accurate approach based on the algorithm presented in Nutsford et al. (2015). This would allow us to tune the sequences of images that represent a cell not only based on the distance to the center, but also considering which photographs better capture the surrounding area.

[1] `https://developers.google.com/maps/documentation/streetview/overview`
[2] `https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1`

Bibliography

Bello, I., Zoph, B., Vaswani, A., Shlens, J., and Le, Q. V. (2019). Attention augmented convolutional networks. *ArXiv*, abs/1904.09925.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1993). Signature verification using a "siamese" time delay neural network. In *Proceedings of the International Conference on Neural Information Processing Systems*, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation policies from data.

Deng, X., Zhu, Y., and Newsam, S. (2018). Spatial morphing kernel regression for feature interpolation. *IEEE International Conference on Image Processing (ICIP)*.

Floriani, L. and Magillo, P. (2003). Algorithms for visibility computation on terrains: A survey. *Environment and Planning B: Planning and Design*, 30(5).

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Hahnloser, R., Sarpeshkar, R., A. Mahowald, M., Douglas, R., and Sebastian Seung, H. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. (2019). Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*.

Huang, G., Liu, Z., and Weinberger, K. Q. (2016). Densely connected convolutional networks. *CoRR*, abs/1608.06993.

Khan, S., Rahmani, H., Shah, S., and Bennamoun, M. (2018). *A Guide to Convolutional Neural Networks for Computer Vision*. Synthesis Lectures on Computer Vision. Morgan & Claypool Publishers.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11).

Leung, D. and Newsam, S. (2010). Proximate sensing: Inferring what-is-where from georeferenced photo collections. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Newsam, S. and Leung, D. (2019). Georeferenced social multimedia as volunteered geographic information. In *CyberGIS for Geospatial Discovery and Innovation*, pages 225–246. Springer.

Nutsford, D., Reitsma, F., Pearson, A., and Kingham, S. (2015). Personalising the viewshed: Visibility analysis from the human perspective. *Applied Geography*, 62.

Qui, S., Achilleas, P., Bozzon, A., and Geert-Jan, H. (2019). Crowd-mapping urban objects from street-level imagery. In *Proceedings of the World Wide Web Conference*.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381.

Seresinhe, C. I., Moat, H. S., and Preis, T. (2018). Quantifying scenic areas using crowdsourced data. *Environment and Planning B: Urban Analytics and City Science*, 45(3):567–582.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sivic and Zisserman (2003). Video google: a text retrieval approach to object matching in videos. In *Proceedings IEEE International Conference on Computer Vision*.

Srivastava, S., Vargas Munoz, J. E., Lobry, S., and Tuia, D. (2020). Fine-grained landuse characterization using ground-based pictures: a deep learning solution based on globally available data. *International Journal of Geographical Information Science*, 34(6):1117–1136.

Srivastava, S., Vargas-Muñoz, J. E., and Tuia, D. (2019). Understanding urban landuse from the above and ground perspectives: A deep learning, multimodal solution. *Remote sensing of environment*, 228:129–143.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L. (2015). The new data and new challenges in multimedia research. *CoRR*, abs/1503.01817.

Workman, S., Souvenir, R., and Jacobs, N. (2017). Understanding and mapping natural beauty. In *IEEE International Conference on Computer Vision (ICCV)*.

Yang, Y. and Soatto, S. (2020). Fda: Fourier domain adaptation for semantic segmentation.

Zhou, S., Wu, J., Wu, Y., and Zhou, X. (2015). Exploiting local structures with the kronecker layer in convolutional networks. *CoRR*, abs/1512.09194.

Zhu, Y., Deng, X., and Newsam, S. (2019). Fine-grained land use classification at the city scale using ground-level images. *IEEE Transactions on Multimedia*, 21(7):1825–1838.