# Automatic Assignment of Geospatial Coordinates to Historical Photos

**Nuno Yatin Manoje Ramanlal**

Thesis to obtain the Master of Science Degree in

## Information Systems and Computer Engineering

Supervisors: Prof. Dr. Bruno Emanuel da Graça Martins
Dr. Patricia Murrieta-Flores

### Examination Committee

Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. Dr. Bruno Emanuel da Graça Martins
Members of the Committee: Prof. João Miguel da Costa Magalhães

**October 2018**

# Acknowledgements

I would like first to thank my thesis advisor Professor Bruno Martins and also João Monteiro for their guidance, availability, and knowledge, which were crucial for the development of my work towards success.

Secondly, I would like to thank my family, and especially my parents for presenting me with all the necessary conditions to carry on my academic journey.

My thanks also go to Instituto Superior Técnico, and Faculty of Science. Both institutions contributed to my personal and academic growth. Additionally, I would like to thank INESC-ID for providing me with a room with a peaceful environment to work on my dissertation.

Lastly, I would like to thank to my friends and colleges who helped me keep motivated during this last year, and keep me company during my late night working journeys.

Nuno Yatin Manoje Ramanlal

# Resumo

A maioria dos dispositivos utilizados para capturar fotografias têm, atualmente, um *GPS* embutido que permite geocodificar as mesmas, de forma instantânea e automática. Na ausência de recursos como o *GPS*, ou para coleções existentes de fotografias antigas, a geocodificação manual pode se tornar numa tarefa longa e exigente. Esta dissertação propõe uma abordagem baseada em *machine-learning* para geocodificar automaticamente fotografias históricas, utilizando para tal redes neurais convolucionais. Estas redes exigem grandes quantidades de dados de treino, e embora as redes sociais e as plataformas de partilha de fotografias correspondam a duas das fontes de dados mais significativas para tarefas de *supervised learning*, não há grandes conjuntos de dados de fotografias antigas geocodificadas disponíveis, o que faz com que a tarefa de geocodificação de fotografias históricas não tenha sido ainda devidamente abordada. No contexto deste trabalho, é apresentada uma rede *end-to-end*, que liga uma rede neural convolucional baseada na arquitetura ResNet para a tarefa de geocodificação automática a uma outra rede convolucional, também baseada na arquitetura ResNet, para realizar transformações em fotos antigas, numa tentativa de aplicar características das fotografias modernas a fotografias históricas. Assim, a rede de geocodificação pode ser pré-treinada com fotos modernas do conjunto de dados do Flickr. Foram realizadas diversas experiências de geocodificação, em diferentes configurações da rede, utilizando coleções existentes de fotografias históricas geocodificadas. Os resultados das experiências mostram que a rede apresentada nesta dissertação é mais eficiente a geocodificar fotografias históricas em comparação a outras redes neurais convolucionais que não utilizam a componente de colorir, destacando assim o potencial desta abordagem.

# Abstract

Most devices used for capture photos have, nowadays, a built-in GPS, and thus they are able to geocode images instantaneously and automatically. In the absence of GPS features, or for existing collections of older photos, manual geocoding can be a long and grueling task. This dissertation proposes a machine-learning based approach to automatically geocode historical photos, leveraging convolutional neural network architectures. These networks require vast amounts of training data, and while social media and photo sharing platforms correspond to two of the most significant sources of data for performing supervised learning, there are not large datasets of old geocoded photos made available, which makes the task of geocoding old historical photos remain not well-covered. It is introduced an end-to-end network, combining a convolutional neural network model based on the ResNet architecture for automated geo-referencing, with another fully-convolutional network, also based on the ResNet architecture, to perform transformations over old photos, in an attempt to resemble the modern ones. This way, the geocoding network can be pre-trained with modern photos retrieved from the Flickr dataset. There were performed several geocoding experiments over different network settings, leveraging existing collections of geo-referenced historical photos. Experiment's results show that the proposed network is more efficient geocoding historical photos in comparison to other evaluated convolutional neural networks that do not use the coloring component, which highlights the potential of this approach.

# Palavras Chave
# Keywords

## Palavras Chave

Geocodificação de fotografias históricas

Análise de fotografias com base em *deep learning*

Arquiteturas de redes neurais convolucionais

Transformações de imagens

## Keywords

Geocoding historical photos,

Deep learning for analyzing photos

Convolutional neural architectures

Image transformations

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Matching a photo to a particular location in the Earth's surface is becoming a conventional and automated process, abstracting any kind of human efforts towards the task of geocoding. Navigational satellites contribute significantly to the automatization of this task, since for the majority of mobile devices or cameras, GPS trackers are built-in features which can instantaneously associate a photo to a location. As a consequence, image sharing services such as Picasa[1] or Flickr (Thomee et al. (2016)) are brimful with accurately geocoded modern photos.

Machine-learning based approaches to automatically geocode photos, specifically Convolutional Neural Networks (CNNs), require vast amounts of training data, and thus, image sharing services make a significant contribution towards this purpose. For instance, several supervised deep learning based methods to geocode modern photos (Weyand et al. (2016), Vo et al. (2017)) were introduced. When considering historical photos, these are not as available in a significant number as the modern ones. Moreover, GPS features were not accessible until recently, so one can not assure the accuracy of the coordinates assigned to the range of old pictures, since the task was most likely performed manually. Nowadays, due to these concerns, the task of using deep learning methods to geocode old photos remains not well covered.

This dissertation addresses the task of automatically geocoding historical photos from New York and San Francisco, leveraging state-of-art methods based on deep neural networks. The conducted experiments in this work were performed on different CNN's architectures, using small pre-existent datasets of historical photos - the Old San Francisco Project dataset[2], and the Old New York Project dataset[3]. Since old photos lack color information and the introduced geocoding networks were pre-trained with modern imagery from the Yahoo Flickr Creative Commons 100 Million dataset (Thomee et al. (2016)), the approach requires the use of image processing techniques. Therefore, the major contribution of this work is the proposal of a novel end-to-end network combining both the CNNs trained to color images and to geocode historical photos.

[1] `http://picasa.google.com`
[2] `http://www.oldsf.org/records.js.zip`
[3] `http://github.com/oldnyc/oldnyc.github.io/blob/master/data.json`

## 1.1 Motivation

Automatically geocoding photos based on their visual contents is a challenging task, since not all photos have unambiguous location clues that may help geocoding them. Even if those clues are present, they can lead to ambiguity, e.g., a picture of the Golden Gate Bridge, in San Francisco, can be mistaken for a photo of the 25 de Abril Bridge, in Lisbon, due to their structural and background scenery similarities. When it comes to old pictures, due to their lack of location information, two potential consequences may arise: 1) photos are shared in Web platforms without any location information, or 2) the location is assigned manually by a user, which may result in inaccurate attribution of geospatial coordinates.

There has been a wide range of preceding work conducted towards the extraction of visual data and the exploration of its potential, however, only few of them attend to old photos. Previous efforts have, for instance, focused on geocoding old San Francisco photos[4] and old New York City photos[5], but the approaches require manual geocoding, which does not guarantee accuracy. Tsioukas et al. (2015) succeeded in building a method for establishing comparisons between the content of historical photos of urban areas and the current day situation. Even though the introduced method thrived in superimpose the present and the past scenery in 3D space, while leveraging a software package to do so, the approach is highly dependent on the existence of historical maps of different time periods. Introducing a fully-automated approach abstracted from human inputs, and which is not reliant on the presence of specific data such as historical maps, can solve two main issues: 1) fix the mistakes made by users while geocoding old photos manually, and 2) geocode historical photos automatically and with a higher accuracy in comparison to the human labeling.

There are plenty of exciting applications for geocoded historical photos, which support the need to cover this field adequately. Accurately geocoded old photos have a crucial role in the tasks of historical reconstructions, studies of landscape changes and evolution, land planning, land cover mapping, among others.

## 1.2 Thesis Proposal

From reasoning about the set of photos to be used in this work, one can conclude that the task implies dealing with two sets of data with very different visual characteristics: old historical

[4] `http://www.oldsf.org`
[5] `http://www.oldnyc.org`

photos to perform evaluations, and modern pictures to pre-train the geocoding networks. A reasonable approach to consider when addressing the proposed task is to factor it in two subtasks: 1) implementing an image transformation technique based on CNNs, and 2) training a model to geocode old photos, also based on CNNs. Either the modern training data is transformed, and the geocoding network is trained with modern photos resembling old ones, or else the old historical photos dataset has to be transformed, in an attempt to make them look like modern ones. Considering the size of both collections, I argue that applying the image transformation CNN in old photos is a more preferable option, allowing the geocoding network to learn from the real color values of modern photos from the large existing datasets.

It is proposed an end-to-end network, combining an image transformation CNN used to color historical photos, with a geocoding network pre-trained with modern data to infer the location of these colored historical pictures. The ResNet50 architecture is leveraged as feature extractor in the coloring network, and it is also used as the geocoding component.

## 1.3 Contributions

The contributions made in this research project are the following:

- Considering the knowledge at hand, it was introduced a novel end-to-end architecture aiming for geocoding historical photos. The end-to-end network combines two networks: the first corresponds to a CNN trained to color grayscale photos, and the second one is another CNN, based on the ResNet50 architecture, pre-trained with modern data from Flickr dataset to geocode photos;

- The introduction of two networks sets, based on the ResNet50 architecture, capable of geocoding both modern and historical photos from the cities of San Francisco and New York. Each set contains networks capable of performing both regression and classification geocoding tasks;

- The comparison of different methods and network configurations concerning the coloring component, such as applying filters to the input pictures, or use different networks to extract high-level features;

- The introduction of a CNN model capable of coloring grayscale images. The model uses the ResNet50 architecture to extract high-level features from the input.

## 1.4 Structure of the Document

The rest of this document is organized as follows. Section 2 presents fundamental concepts related to the task, and previous related work concerning image processing methods and networks, image transformations techniques, and methods for automatic image geocoding. Section 3 details the approach and the architecture used in the implemented components of the end-to-end network. Section 4 reports the results of the experiments conducted on the trained neural networks. Finally, Section 5 outlines and summarizes the main findings of this work, and refers to possible future developments to be done within the field.

# 2 Concepts and Related Work

This chapter is divided into two sections. Section 2.1 overviews the main underlying concepts regarding modern machine learning methods for the task of image processing, which this work relies on. Section 2.2 presents introduced neural networks for the task of image processing, and reviews previous studies addressing image handling, specifically methods to transform and geocode them.

## 2.1 Fundamental Concepts

The growth of computational resources was followed by the emergence of more robust and efficient techniques for pattern recognition tasks. Next, there are described some of those techniques, specifically both the single and multi-layer perceptrons, followed by a description of CNNs. At last, there are presented some activation functions used in neural networks.

### 2.1.1 Single-layer and Multi-layer Perceptrons

Artificial neural networks are nowadays commonly used for image processing. They typically consist of several simple processing units wired together in a network. Each unit is configured to act like a neuron, being activated depending on the strength of the input signal received from other nodes.

The single-layer perceptron is perhaps the simplest neural network, which is used for supervised learning of binary classifiers, i.e., the perceptron classifies linearly separable patterns with a binary target. This model makes predictions based on a function which uses a set of weights to linearly combine the inputs given as a feature vector – see Figure 2.1. Besides the input, the bias allows a classifier to shift the decision boundary according to prior information on the class frequency. The weights define the slope of the linear prediction function. Given these parameters, the weighted sum $\alpha$ can be defined as follows:

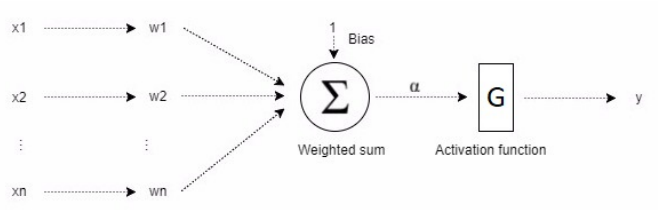


Figure 2.1: Illustration of the perceptron classification model.

$$\alpha = \left( \sum_{i=1}^{n} w_i \times x_i \right) + b \tag{2.1}$$

In the previous equation, $n$ is the dimensionality of the input vector $\mathbf{x}$, while $\mathbf{w}$ is the vector of weights and $b$ is the bias term. The activation function $G$ takes the linear combination produced by $\alpha$ as input, and given a threshold $\theta$, (e.g., the value zero) produces the output classification.

$$\mathrm{G}(\alpha) = \begin{cases} 1 & \text{if } \alpha \geq \theta \\ -1 & \text{otherwise} \end{cases} \tag{2.2}$$

For each training instance, after checking the output, the error is calculated as follows:

$$E = \mathrm{G}(\alpha) - \text{actual output} \tag{2.3}$$

Then, the bias $b$ and the weight vector are updated according to a learning rate $r$:

$$b = b + r \times E \tag{2.4}$$

$$\forall i \in n : w_i = w_i + E \times r \times x_i \tag{2.5}$$

Each complete run through all instances of the input is called an epoch. The learning process involves iterating through the data in epochs until a certain convergence criterion is met, such as a maximum number of iterations or if the error becomes equal 0 in all the instances.

The multilayer perceptron was later introduced to consider the classification of non linearly separable data. It is based on the idea of having several perceptrons organized into layers: an input layer, an output layer, and one or more hidden layers. The learning process is based on the backpropagation algorithm, which has two phases: 1) in the forward phase, for each neuron

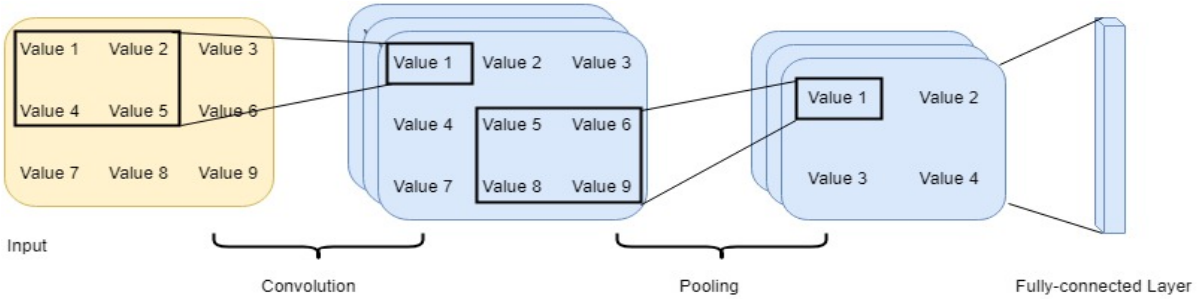


Figure 2.2: Illustration of the CNN architecture.

in each layer, the weighted sum $\alpha$ is calculated using Equation 2.1, passed onto an appropriate activation function, e.g. a sigmoid function, and the resulting value determines the neuron's output, which becomes the input value for the neurons in the next layer connected to it; 2) in the backward phase, the error defined in the output layer is propagated backward throughout the perceptron. The error $E$ in the $k$-th node of the output layer is defined as follows:

$$E = (t_k - O_k) \times O_k \times (1 - O_k) \tag{2.6}$$

In Equation 2.6, $O_k$ is the value $k$, and $t_k$ is the target output of the node. $O_k \times (1 - O_k)$ represents the derivative of the sigmoid activation function. Concerning the hidden layers, to modify the weight $w_{j,k}$ between the output node $k$ and the node $j$, one should use the following expression:

$$w_{j,k} = w_{j,k} + (l_r \times \delta_k \times x_k) \tag{2.7}$$

In the previous equation, $(l_r \times \delta_k \times x_k)$ represents the change rate in the weight between nodes $j$ and $k$, while $\delta_k$ is the value of the sigmoid activation in node $k$, $x_k$ is the input value of the node $k$, and $l_r$ is the learning rate.

### 2.1.2 Convolutional Neural Networks

Previous studies have shown that multilayer perceptrons struggle with the computational complexity required to process image data. A common alternative used in image processing tasks are CNNs, which correspond to a variation of multilayer perceptrons designed to require minimal processing through a shared weights architecture - Rawat and Wang (2017). These networks take as input images and the layers within them are comprised of neurons arranged into three dimensions: height, width, and depth. The neurons within any given layer will only

connect to a small region of the layer preceding it. As shown in Figure 2.2, typical CNNs are commonly comprised of three types of layers: convolutional layers, pooling layers, and fully-connected layers.

The convolutional layer's parameters consist of a set of learnable filters. Usually, filters are spatially small but extend through the full depth of the input volume, producing a 2-dimensional activation:

- Each filter is applied across the width and height of the input;
- For every position $x$ of the input:
  - Dot products are computed between the entries of the filter and the input, generating a matrix $m$;
  - The values in $m$ are summed, and then divided by the number of values in $m$, generating a single value $v$;
- Assign to the position $x$ the value $v$.

For each filter, there will be an activation map. The maps are then stacked to form the full output volume from the convolutional layer. Every neuron is only connected to a small region of the input volume, and the dimensionality of that region is referred to as the receptive field size.

The pooling layer uses an aggregation operation (e.g., the maximum or the average) to progressively reduce the spatial size of the representation, so that the number of parameters in the network is also reduced. This layer is usually placed in-between successive convolutional layers. If the volume size input is $W_1 \times H_1 \times D_1$, and considering $F$ and $S$ as spatial extent and stride, respectively, the volume produced has size $W_2 \times H_2 \times D_2$, where:

$$W_2 = (W_1 - F)/S + 1$$
$$H_2 = (H_1 - F)/S + 1$$
$$D_2 = D_1$$

Stride is used in convolutional layers to define the number of input cells (e.g., pixels in an image) the filter must move at a time. For instance, if the stride is 2 then filters jump 2 pixels at a time as they are being slid around the input.

The last layer before the output, i.e. the fully-connected layer, contains neurons with a full connection to all activations in the previous layer. This layer will attempt to produce the desired outputs from the previous activations.

### 2.1.3 Activation Functions

Activation functions are essential element-wise operations in layered architectures, as they take a crucial role in generating the aforementioned activation maps. The convolutional layer can be implemented along with a Rectified Linear Unit (ReLU), which is a popular non-linear activation function in CNNs, and one of the simplest:

$$f(x) = \max(0, x) \tag{2.8}$$

One other very used activation function is the softmax. Like the sigmoid function, the softmax function squashes each unit's output between 0 and 1, but it also divides each output such that the total sum of the output values is 1. Given a vector $\boldsymbol{z}$, for each element $j$ in the vector the softmax can be computed as follows:

$$\sigma(\boldsymbol{z})_j = \frac{e^{\boldsymbol{z}_j}}{\sum_{k=1}^{K} e^{\boldsymbol{z}_k}} \tag{2.9}$$

While ReLU is a function usually applied between convolutional layers, the softmax function is applied in the final layer of a network and used to make classification.

Ramachandran et al. (2017) proposed to seek for function as an alternative to the ReLU. The experiments led to the recognition of a new activation function called Swish. The introduced function was tested across several challenging datasets, performing better than ReLU on deeper models. Swish is defined as follows:

$$f(x) = x \cdot \sigma(\beta x) \tag{2.10}$$

In the previous equation, $\beta$ can be either a constant or a trainable parameter, and $\sigma$ is the sigmoid function, defined as follows:

$$\sigma(z) = (1 + e^{-z})^{-1} \tag{2.11}$$

## 2.2 Related Work

The following sections describe previous studies concerning image transformation tasks, as well as machine learning methods for image analysis and recognition, including studies focusing on image geocoding. Section 2.2.1 introduces several methods and networks for image processing, Section 2.2.2 presents methods to perform image translation and style transferring, and finally Section 2.2.3 reports applications of machine learning methods in the field of image geocoding.

### 2.2.1 Deep Learning Methods for Image Classification

This section reports several deep learning methods that support the premises that machine learning techniques applied to neural networks have been essential in the design of efficient object recognition systems. One of the first, and best known, successful application of convolutional networks was introduced by LeCun et al. (1998). In a case study regarding character recognition, the authors showed the advantage of applying machine learning techniques over hand-crafted feature extractors. At the time, pattern recognition was performed using a fixed feature extractor that would extract the input's relevant information and build feature vectors, and a trainable classifier that would convert those vectors into classes.

Aiming for building a feature extractor that relies on automated learning as much as possible, the authors introduced the LeNet convolutional network. The architecture is made of six trainable layers and takes a $32 \times 32$ image of characters. Given an input data, the network first applies convolutional operations and subsampling operations alternatively. Then, the resulting representation of data is fed into a fully connected neural network to finalize the classification task. There are three convolutional layers to perform the convolutional operations: $C1$, $C3$, and $C5$, and two layers in charge of subsampling operations: $S2$, and $S4$. These two layers are responsible for reducing the size of data, thus, reduce computation complexity. $F6$ corresponds to the fully-connected layer, which is fully connected to $C5$, containing 84 units. The output of each unit $y_i$ of the output layer is computed as follows:

$$y_i = \sum_j (\boldsymbol{x}_j - \boldsymbol{w}_{i,j})^2 \tag{2.12}$$

In the previous equation, $\boldsymbol{x}$ represents the input vector and $\boldsymbol{w}$ the parameter vector.

To improve the performance of object recognition, Krizhevsky et al. (2012) introduced and trained a deeper CNN than LeNet, named AlexNet, using a larger dataset, named ImageNet

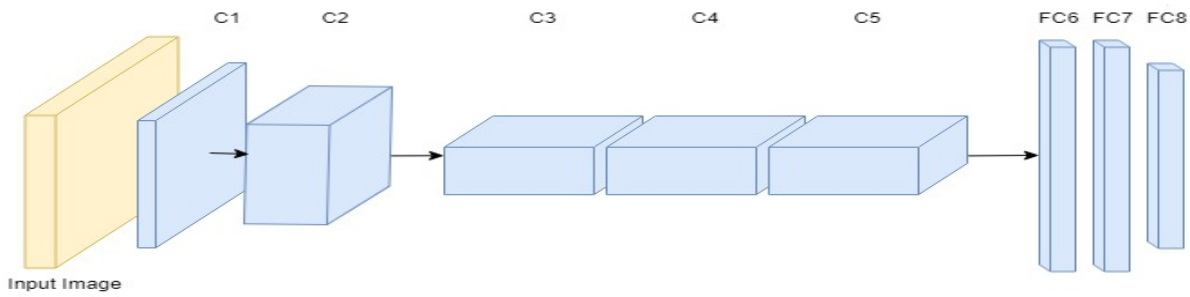


Figure 2.3: Illustration of the AlexNet CNN architecture.

(Russakovsky et al. (2015)), and using techniques to avoid overfitting. The network outperformed the previous state-of-the-art methods, achieving the top-1 and top-5 error rates in the ImageNet LSVRC-2010 contest[1].

AlexNet is a CNN architecture capable of performing image classification and detection. It contains eight layers: five convolutional layers and three are fully-connected layers − see Figure 2.3. The output of the last fully-connected layer is applied to a softmax activation, producing a distribution over the target class labels. The network has ReLU activations attached to it, after every convolutional and fully-connected layer.

In order to learn the 60 million parameters of the network architecture without overfitting, the authors implemented two methods: data augmentation and dropout. The first technique of data augmentation involves extracting, randomly, $224 \times 244$ patches from the training images of dimensions $256 \times 256$, increasing the size of the set by a factor of 2048. The second technique of data augmentation involves changing the intensities of the RGB values in the training images, performing Principal Component Analysis (PCA) on the pixels. For each training instance, the authors added multiples of the principal components found. Given an RGB image pixel $I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T$, the authors added the following value: $[p_1, p_2, p_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$, where $p_i$ and $\lambda_i$ are the $i$-th eigenvector and eigenvalue of the covariance matrix of RGB pixel values respectively, and $\alpha_i$ is a random value.

The second approach used by the authors to reduce overfitting was dropout. At each training stage, the output of each hidden neuron with a probability of 0.5 is set to zero. The dropped out neurons do not contribute neither to the forward or to the backpropagation of the error.

Based on the LeCun and AlexNet approaches and results, one can conclude that using larger networks leads to an improvement of the outputted results. Simonyan and Zisserman (2014) confirmed the importance of the convolutional network depth in pattern recognition

[1] `http://www.image-net.org/challenges/LSVRC/2010/`

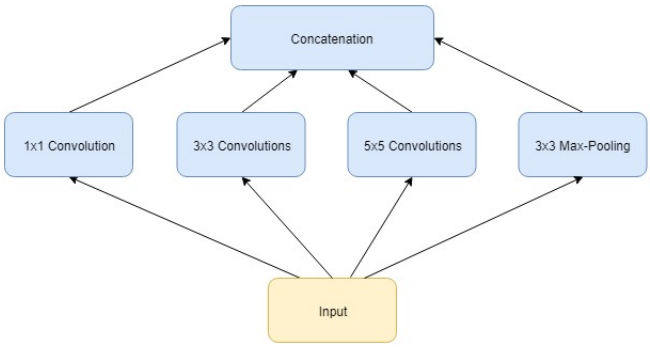


Figure 2.4: Illustration of the Inception module for image classification.

tasks, performing several evaluations over a deeper network, the VGG network. Adding more convolution layers led to a significant growth in accuracy.

The VGG network takes as input fixed-size $224 \times 224$ RGB images, which are passed through a stack of convolutional layers and reduced receptive field size filters. There are also five max-pooling layers, which follow some of the convolutional layers. Max-pooling is performed over a $2 \times 2$ pixel window, with a stride value of 2. After the convolutional layers, there are three fully-connected layers, all presented with ReLU non-linearity.

The authors performed a set of evaluations over the VGG network, by testing different model configurations over the architecture. All the configurations follow the structure described above and differ only in the depth. There were performed two types of evaluations: single test scales evaluation and multiple test scales evaluation. In the single scale evaluation, the authors noticed an error decreasing with the increase of the depth: from 11 layers to 19 layers, and once the depth is in 19 layers the error saturates. Regarding the multiple scales evaluation, the authors tested the same image with several rescaled versions of it over the models, with the deepest configurations - 16 and 19 layers - performing the best.

As the networks became more complex and deeper, they start requiring more computation efforts. With the emergence of the already mentioned Web 2.0, having efficient algorithms for mobile vision applications start becoming a need. Szegedy et al. (2015) was able to present a convolutional architecture with an increased depth and width, focusing in maintaining a sustainable computational budget, denominated Inception. This network defined a new state of the art for classification and detection in a challenge that evaluated algorithms for object detection and image classification at large scale[2].

The idea implemented in these convolutional networks is to analyze correlation statistics

[2] `http://www.image-netorg/challenges/LSVRC/2014/`

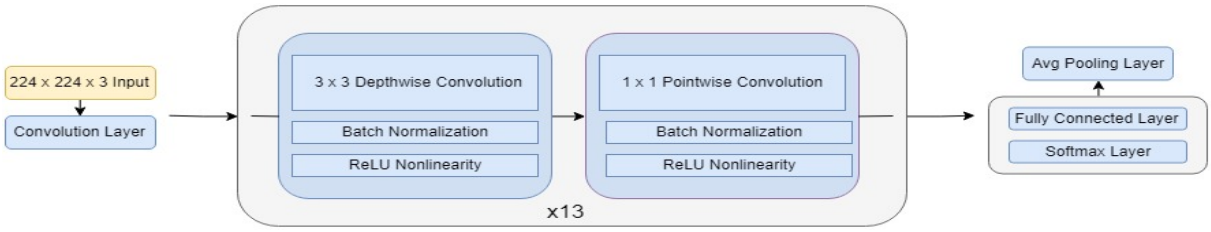


Figure 2.5: Illustration of the MobileNet's architecture.

between units from the previous layer of activations, grouping the highly correlated ones for the next layer. Taking advantage of the fact that the hidden layers near the input layer contains high correlations in image patches, those are covered by $1 \times 1$ convolutions, while $3 \times 3$ and $5 \times 5$ convolutions are also used to filter the input along with the first. The architecture, shown in Figure 2.4, is a combination of all three convolutions, as input to the next stage, together with a pooling layer.

Using multiple convolutions on a single patch allow the visual information to be processed at several scales, and then aggregated for the next stage. To make this process less computationally expensive, and to avoid an increase of output numbers between stages, the authors introduced an alternative model that applies $1 \times 1$ convolutions to reduce the dimensionality of the input image, and only then $3 \times 3$ and $5 \times 5$ convolutions are performed. In the end, all three convolutions are concatenated together with the max-pooling operations.

Still concerning the efficiency of complex neural networks, Howard et al. (2017) presented a set of efficient and small network architectures, called MobileNet, to be used in mobile and computer vision applications. These networks are mainly based on depthwise separable convolutions, which is a technique that consists in performing, independently, a spatial convolution over each channel of a given input, followed by a $1 \times 1$ convolution, referred to as pointwise convolution, that creates a linear combination of the output of the spatial convolutions, generating new features.

The MobileNet structure uses $3 \times 3$ depthwise separable convolutions, resulting in 8 to 9 less computation required comparing to standard convolutions, and all the layers are followed by batch normalization and ReLU nonlinearity, apart from the first convolution and last fully-connected layers. Figure 2.5 illustrates the network's architecture, in which one can notice that the depthwise separated convolutions layers are sequentially repeated thirteen times.

Xception is yet another CNN proposed by Chollet (2017), which is entirely based on depthwise separable convolution layers. The authors proposed to replace Inception models with stacks of depthwise separable convolutions. While both models share the same number of parameters

the Xception does a more efficient use of them. Moreover, contrarily to the Inception module, depthwise separable convolutions are usually implemented without non-linearities such as ReLU. The Xception architecture has 36 convolutional layers, all linearly stacked with residual connections (He et al. (2016)).

Both Inception V3 and the Xception model were compared during the experimental evaluation. Inception V3 corresponds to the third version of the Inception model, and it was introduced by Szegedy et al. (2016). The comparison was conducted on the 1000-class single-label classification task of the ImageNet dataset (Russakovsky et al. (2015)) and in the 17000-class multi-label classification task of the JFT dataset (Hinton et al. (2015)).

Concerning classification performance on the ImageNet dataset, Xception showed marginally better results than Inception V3. On JFT, Xception showed a 4.3% relative improvement on the FastEval 14k MAP@100 metric, which is a metric used to evaluate the performance of the FastEval 14k dataset, based on mean average precision. The Xception architecture showed a much larger performance improvement on the JFT dataset compared to the ImageNet dataset, and the authors believe this is because Inception V3 was developed with a focus on ImageNet. Overall, when using the ImageNet dataset, the Xception does not outperform significantly the InceptionV3, whereas it shows large gains on the JFT dataset.

### 2.2.2 Methods for Performing Image-to-Image Transformations

The following subsections present methods used for performing different types of image transformations. Section 2.2.2.1, presents an approach for learning to translate images in the absence of paired examples. Section 2.2.2.2 and Section 2.2.2.3, refer to methods for style transferring, i.e., given an input image and a referenced style image, it is produced a new image matching the input image but with a new style applied to it.

Besides the studies presented in the following subsections, there are several other methods that aimed to transform images taking an image-to-image translation approach, such as the one presented by Choi et al. (2017), which allows performing transformation from a single image to multiple domains. There are also several other coloring approaches such as the ones introduced by Zhang et al. (2016), Larsson et al. (2016), and by Iizuka et al. (2016). All three methods leverage deep networks to perform fully automatic coloring, however using different network architectures and loss functions.

#### 2.2.2.1 Unsupervised Image-to-Image Translation Networks

Given two domains, image-to-image translations consist in mapping two images from different domains. When there are not paired examples of translated photos, this task is referred to as unsupervised image-to-image translation.

Zhu et al. (2017) presented an approach capable to capture characteristics from an image collection and translate those into a different image collection, without paired examples. To do so, the authors learned a mapping between domains $X$ and $Y$, $G : X \rightarrow Y$, with the output $\gamma = G(x)$, $x \in X$ being indistinguishable from images $y \in Y$. To ensure that an input $x$ is correctly paired with an output $y$, the authors also learned an inverse mapping $F : Y \rightarrow X$, under the same conditions as the previous mapping function. Additionally, they formulate two adversarial discriminators: $D_x$ distinguishes between images $x$ and translated images $F(y)$, whereas $D_y$ make distinctions between images $y$ and translated images $G(x)$. These two adversarial discriminators were applied in the already mentioned mapping functions.

Considering the mapping $G : X \rightarrow Y$ and the corresponding discriminator $D_y$, the objective function is described as follows:

$$\zeta_{GAN}(G, D_y, X, Y) = E_{y \sim p_{data(y)}}[\log D_y(y)] + E_{x \sim p_{data}(x)}[\log(1 - D_y(G(x))] \tag{2.13}$$

Function $G$ is in charge of generating images $G(x)$ similar to the ones from domain $Y$. The discriminator $D_y$ ensures that the images $G(x)$ and $y$ are indistinguishable. This objective function was also applied for the second mapping function $F : Y \rightarrow X$ and the corresponding discriminator $D_x$.

Given that the approach consists in mapping photos from one domain to another, in order to reduce the number of possible matches, i.e., the space of possible mapping functions, the authors introduced cycle-consistent mapping functions. In the forward cycle consistency, given a translated image $G(x)$, the image translation cycle should map it back to the original image $x$, i.e., $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. The same idea is applied in the backward cycle consistency.

The full objective is defined as follows:

$$\zeta(G, F, D_x, D_y) = \zeta_{GAN}(G, D_y, X, Y) + \zeta_{GAN}(F, D_x, Y, X) + \lambda\zeta_{cyc}(G, F) \tag{2.14}$$

In Equation 2.14, $\lambda$ controls the relative importance of the objectives, and $\lambda\zeta_{cyc}(G, F)$ refers to the cycle consistency loss.

In the implementation phase, the authors leveraged a network based on the one introduced by Johnson et al. (2016), adapting it to contain two stride-2 convolutions and several residual blocks, specifically 6 blocks for $128 \times 128$ images, and 9 blocks for $256 \times 256$.

In a case study, the authors evaluated their method against several quantitative and qualitative baselines, using the same evaluation datasets and metrics as Isola et al. (2016).

- AMT perceptual studies: This metric was used in a map↔aerial task, in which the authors ran *real versus fake* perceptual studies to verify the realism of outputs. Each one of the 25 participants saw a sequence of pairs of real-fake images, and they were asked to select image the they considered the real one.
- FCN score: This metric was used in a labels→photo task, using the Cityscapes dataset (Cordts et al. (2016)). The FCN predicts labels for a given photo, therefore in the presence of a photo from a label map of *car on road*, the FCN must detect *car on road*.
- Semantic segmentation metrics: These metrics were used in the photo→labels task associated to the Cityscapes benchmark, which includes, among others, per-pixel accuracy and per-class accuracy.

Some of the comparative baselines used by the authors include CoGAN (Liu and Tuzel (2016)), SimGAN (Shrivastava et al. (2017)), and pix2pix (Isola et al. (2016)). All the baselines were implemented using the same architecture. The results were not remarkable with any of the unsupervised baselines. However, the author's method can produce translations of similar quality to pix2pix. In another case study, the authors verified that removing the cycle-consistency loss significantly harm the results.

The method was also demonstrated in several applications, such as generating photos in several styles. After training the model using landscape photographs downloaded from Flickr[3] and WikiArt[4], it is noticeable that the method can learn to represent the style of entire collections. Overall, the success of the method is depending on the task. While results in many cases are good, in tasks that require, for instance, geometric transformations, the model does not perform well.

#### 2.2.2.2 Deep Photo Style Transfer

Luan et al. (2017) introduced a deep-learning approach, based on a previous neural style

[3] `http://www.flickr.com/`
[4] `http://www.wikiart.org/`

transfer approach by Gatys et al. (2016), for photographic style transfer. The Gatys's approach was extended in this work with a photorealist regularization term in the objective function to prevent color distortions, and an existing method to avoiding content-mismatch problems. The algorithm used to perform the task applies the style of a reference image $S$ onto an input image $I$, producing a second version of the input image $O$, with the new style applied on it. This is done my minimizing the following function:

$$\zeta_{total} = \sum_{\ell=1}^{L} \alpha_\ell \zeta_c^\ell + \Gamma \sum_{\ell=1}^{L} \beta_\ell \zeta_s^\ell \tag{2.15}$$

In the previous equation, $L$ is the total number of convolutional layers and $\ell$ indicates the $\ell$-th convolutional layer of the deep CNN. The parameters $\alpha_\ell$ and $\beta_\ell$ are the weights to configure layer preferences, and $\Gamma$ is a weight that balances the tradeoff between the content and the style of the image.

In order to produce photorealistic outputs by preserving the structure of the input image, the authors assumed that the input photo is already realistic and added a term to penalize image distortions during the transformation of it. Moreover, to avoid painting-like effects, the author's approach only applies color-wise transformations. The penalty equation was based on the Matting Laplacian (Levin et al. (2008)):

$$\zeta_m = \sum_{c=1}^{3} V_c[O]^T M_I V_c[O] \tag{2.16}$$

In the previous equation, $V_c[O]$ is the vectorized version of the output image $O$ in channel $c$, and $\boldsymbol{M_I}$ is the matrix of the input image $I$ with $N$ pixels.

To address the problem of the variations of semantic context which causes wrong mappings between elements, (i.e., the sky in the input photo may be matched to the sea in the reference images) the authors proposed generating image segmentation masks for the input and reference images, regarding labels such as the sky and buildings. The masks were added to the input image as additional channels thus updating the style loss:

$$\zeta_{s+}^l = \sum_{c=1}^{C} \frac{1}{2N_{l,c}^2} \sum_{ij} (G_{l,c}[O] - G_{l,c}[S])_{ij}^2 \tag{2.17}$$

In the previous equation, $C$ is the number of channels in the semantic segmentation mask and $c$ is a specific channel of the segmentation mask in layer $l$.

The final formulation of the photorealistic style transfer objective was defined as follows:

$$\zeta_{total} = \sum_{\ell=1}^{\boldsymbol{L}} \alpha_\ell \zeta_c^\ell + \Gamma \sum_{\ell=1}^{L} \beta_\ell \zeta_{s+}^\ell + \lambda \zeta_m \tag{2.18}$$

The authors compared their method with previous approaches, using both indoor and outdoor scenes. Contrarily to previous works, where the output contains painting-like distortions, the author's method produced better results because of the photorealism regularization and semantic segmentation. However, there were still content-mismatches problems in some cases, which can be fixed using manual segmentation.

Two user studies were also conducted. In the first study, the authors evaluate the photorealism of methods such as CNNMRF (Li and Wand (2016)) and neural style (Gatys et al. (2016)), against their technique. The users were asked to classify 32 images using a 1-to-4 range, from *definitely not photorealistic* to *definitely photorealistic*. The results showed that the author's approach only scored below histogram transfer. The second study was conducted to estimate the efficiency of the style transfer techniques. Users were shown a style image and four transferred outputs, being one of them from the author's method. They were then asked to choose the most similar image to the reference style image. The results obtained from the author's algorithm were considered the most similar to the reference style in 80% of the cases.

#### 2.2.2.3 Fast Patch-based Style Transfer of Arbitrary Style

Chen and Schmidt (2016) introduced a style transfer procedure not limited to a finite set of styles, which makes use of a single layer of a pre-trained network to combine both the input image structure and the reference style. One of the most important procedures to perform the style transfer is referred to as swapping the style. It is a patch-by-patch operation, so given the RGB representations $\boldsymbol{C}$ and $\boldsymbol{S}$ of the content and style images respectively, and given the function $\Phi$ that maps the RGB values of an image to some intermediate activation space, the style swap procedure works as follows:

- There are extracted the patches $\{\Phi_i(\boldsymbol{C})\}_{i \in n_c}$ and $\{\Phi_j(\boldsymbol{S})\}_{j \in n_s}$ corresponding to the input and style activations, where $n_c$ and $n_s$ are the number of extracted patches;

- It is determined the closest style patch, for each content patch. This is done based on the normalization cross correlation measure $\Phi_i^{ss}(\boldsymbol{C}, \boldsymbol{S})$, for each patch. The correlation

measure is defined as follows:

$$\Phi_i^{ss}(\boldsymbol{C}, \boldsymbol{S}) = \underset{\Phi_j(S), j=1,\ldots,n_s}{\operatorname{argmax}} \frac{\langle \Phi_i(C), \Phi_j(S) \rangle}{|\Phi_i(C)| \cdot |\Phi_i(S)|} \tag{2.19}$$

- Each content activation patch $\phi_i(\boldsymbol{C})$ is swapped with the determined closest-matching style patch $\Phi_i^{ss}(\boldsymbol{C}, \boldsymbol{S})$;
- After the swapping procedure, the activations can be reconstructed.

Still regarding the swap operations, the authors proposed an efficient implementation of the procedure using a network with three operations: a 2D convolutional layer, a channel-wise argmax, and a 2D transposed convolutional layer. The author's implementation can be described through the following steps:

- Using the style activations patches as convolution filters, and $\Phi(C)$ as input, a tensor $K$ is computed, with $n_c$ spatial locations and $n_s$ feature channels;
- Each spatial location $K_{a,b}$ is replaced by a one-hot vector corresponding to the best matching style activation patch $\bar{\mathrm{K}}$;
- A 2D transposed convolution with style activation patches $\{\Phi_j(\boldsymbol{S})\}$ as filters is applied to the input $\bar{\mathrm{K}}$. At each spatial location, only the best matching style activation patch is in the output.

Once the style transfer is done, an inverse network is in charge of generating the final image. In sum, given an input image, it is first processed by a CNN, then the style swap procedure is applied, and finally an inverse CNN is used as a decoder, to generate the output image with the new style. This method is referred to as the feed-forward method, and it can be resumed in the following three steps:

- Compute $\Phi(\boldsymbol{C})$ and $\Phi(\boldsymbol{S})$;
- $\Phi^{ss}(\boldsymbol{C}, \boldsymbol{S})$ by style swapping;
- Feed $\Phi^{ss}(\boldsymbol{C}, \boldsymbol{S})$ into a trained inverse network.

For evaluation, the authors analyzed the effects of the style swapping in different layers of the VGG-19 Simonyan and Zisserman (2014) network and concluded that style swapping on the

"ReLU 3_1" layer, which is the first ReLU layer after the second max-pooling layer of the VGG-19 architecture, produces the most acceptable results without deforming the content's structure of the input image.

### 2.2.3 Methods for Automatic Image Geocoding

The following subsections present methods for automatic image geocoding. The first subsection presents a previous study that addressed the geocoding task as a classification problem. Next, it is described an approach in which the authors proposed to use deep CNNs to address the problem of cross-view image geolocalization. Finally, it is described an approach to geolocate images, combining the deep image classification solution from Section 2.2.3.1 with an approach named Im2GPS, which relies on similarity search.

Besides the approaches presented in the following subsections, there are many other documented studies that tackled the challenging task of estimating the location where a photo was taken. Johns et al. (2017) described two classification approaches to geocode photos. The first approach only leverages the image content and is similar to the one described in Section 2.2.3.1. Contrarily to PlaNet, the Earth's surface was divided into cells using a Delaunay triangle-based meshing architecture. The second model was trained to perform the geocoding task using both the content and time metadata of the image, showing that incorporating additional data to the model can increase the accuracy. Zhu and Bain (2017) introduced a variant of this classification method, named Branch Convolutional Neural Network (B-CNN), which outputs multiple predictions considering a hierarchical structure of target classes. A B-CNN architecture contains multiple output branch networks in between the building blocks of the CNN. The internal outputs of those building blocks are used to build a hierarchical prediction. Each output branch contains fully connected layers and a softmax layer to process the internal output from a specific part of the CNN. For instance, in a B-CNN an image of a mouse will contain a hierarchical label of [natural, small, mammals, mouse].

#### 2.2.3.1 Photo Geolocation with Convolutional Neural Networks

Weyand et al. (2016) proposed a classification-based model for image geocoding. To extend the localization of a single photo to an entire album, the proposed model was combined with a Long Short-Term Memory (LSTM) approach. Using an LSTM increase accuracy by reducing ambiguity, i.e., a croissant photo could be taken anywhere in the world, but if it is on the same

album as a photo of the Eiffel Tower, the LSTM model will use this cue to geolocate it to somewhere within the city of Paris.

To build the target classes of the classification model, the authors subdivided the Earth's surface into a set of geographical cells. To do so, it was used the Google's open source S2 geometry library[5]. This library projects the surfaces of an enclosing cube on the Earth's surface. Each side of the cube is subdivided hierarchically by six quad-trees, therefore this technique defines a hierarchical partitioning of the globe. To avoid an imbalanced class distribution, the authors kept dividing the cells recursively untill no cell contained more than a fixed number of photos. This partition criterion made sparsely populated areas being covered by larger cells, whereas densely populated areas were covered by smaller cells. The model does not consider locations where photos are not likely to be captured, since cells containing less than a certain minimum number of photos are removed.

The CNN model to perform the geocoding task was based on the Inception architecture, introduced by Szegedy et al. (2015), with a softmax layer assigning a probabilistic value to each geographical cell generated from the partition. The weights of the model were initialized randomly, and the authors used an initial learning rate of 0.045. The training dataset contains 126 million photos.

To measure the geolocation accuracy, 2.3 million new photos were extracted from Flickr. The localization error was computed by measuring the distance between the center of the predicted cell to the original location of the photo. This measure may lead to incorrect interpretations since even if the ground truth location corresponds to the predicted cell, the error can still be large depending on the cell size. In order to compare the model performance with human performance, the authors made a test through which they concluded the model outperforms humans by a considerable margin. The model also outperformed the state-of-the-art local feature based image retrieval approach from Sharif Razavian et al. (2014), on a test dataset introduced by Jegou et al. (2008).

Regarding now the task of geocoding an entire album, the authors proposed to use a neural LSTM architecture (Hochreiter, Sepp and Schmidhuber, Jürgen (1997)). LSTM networks have as default behavior the capacity to remember information for long periods of time. These networks have a form of a chain of repeating modules (or blocks) of a neural network. Each block contains gates to control the information that comes in at time $t$, i.e., the new memory component vector $\boldsymbol{x_t}$ and the hidden state component vector $\boldsymbol{h_{t-1}}$. The forget gate, defined by

[5] `http://code.google.com/p/s2-geometry-library/`

the sigmoid function $\mathrm{f_t}$, decides which information to forget, as shown below:

$$\mathrm{f_t} = \sigma(\boldsymbol{W_f} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + b_f) \tag{2.20}$$

The input gate, defined by the sigmoid function $\mathrm{i_t}$ shown below, defines which values will be updated. Then, $\boldsymbol{O_t}$ corresponds to a vector of new candidate values to be added to the memory cell. Finally, Equation 2.23 updates the old memory cell $\boldsymbol{C_{t-1}}$ into $\boldsymbol{C_t}$:

$$\mathrm{i_t} = \sigma(\boldsymbol{W_i} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + b_i) \tag{2.21}$$

$$\boldsymbol{O_t} = \tanh(\boldsymbol{W_c} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + b_C) \tag{2.22}$$

$$\boldsymbol{C_t} = \mathrm{f_t} \times \boldsymbol{C_{t-1}} + \mathrm{i_t} + \boldsymbol{O_t} \tag{2.23}$$

The output gate, defined by the sigmoid function $\mathrm{o_t}$, and shown next, is used to decide the parts to be outputted. After applying the sigmoid function to the memory cell, the cell goes through a hyperbolic tangent activation function to output only the desired parts.

$$\mathrm{o_t} = \sigma(\boldsymbol{W_o} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + b_o) \tag{2.24}$$

$$h_t = \mathrm{o_t} \times \tanh(\boldsymbol{C_t}) \tag{2.25}$$

In a first model, given an image, the authors extracted the output vector from the layer before the softmax layer in PlaNet, and fed it into the LSTM units. The output of the LSTM units is again fed to the geocoding network, so that the softmax layer can generate the classification output. Many albums have several images at the beginning containing no relevant visual clues. Since the model is unidirectional, once the LSTM units retain information about the initial instance of the album, the judgment of the remaining instances will be affected from the previously acquire knowledge. Thus, the authors introduced the second model.

In a second model, the authors postponed the inference, avoiding making predictions based on those images in the beginning with no location clues. This way, the model only starts making predictions after it accumulates information from several images. However, the results showed that these modifications do not improve localization accuracy. The author's assumption is that since the mapping from the input image to output labels became more complex, the prediction became more difficult. Moreover, if one starts making predictions after seeing 2 pictures, but the first image with high location confidence occurs only after 3 steps, the prediction for the first image will probably be still be wrong.

A third model makes predictions using a two steps procedure. First, the model goes throw all images of an album, and only after seeing all photos, it starts making predictions. This approach outperforms the previously introduced LSTM models, but since the model learns to rely on its previous predictions, if there is a low-confidence image at the beginning of the album sequence, that image will be assigned to the last confident location in the sequence, which in this case corresponds to the end of the sequence.

A last model used by the authors was a bi-directional LSTM, It consists in the combination of two LSTM models: the first one processes the input sequence from left to right, and the second one processes the sequence from right to left. The evaluation showed that BLSTMs outperformed repeated LSTMs, solving the previously verified problems.

#### 2.2.3.2 Wide-Area Image Geolocalization with Aerial Imagery

Some works address the task of geocoding photos, using their visual information and matching them against a dataset of ground-level images based on similarity (Hays and Efros (2008)). However, in cases where there are no ground-level images of certain locations, these methods fail. Leveraging the significant number of high-resolution aerial images, Workman et al. (2015) proposed to use deep CNNs and map ground-level images to the corresponding aerial images. To perform the task, the authors introduced a cross-view training, which consists in leveraging existing CNNs trained to extract ground-level image features and use them to learn to extract the same features from aerial images.

The authors used deep feed-forward neural networks as features extractions functions $f_a$ and $f_g$. $f_a(l;\theta_a)$ extracts features from the aerial imagery with a center location $l$, whereas $f_g(I;\theta_g)$ extracts features from a ground-level image $I$. Both $\theta_a$ and $\theta_g$ are parameters for feature extraction, and they both include the network architecture and the weights. To train a single-scale model, the goal is to minimize the following objective function:

$$\mathrm{J}(\theta_a) = \sum_i |f_a(l_i;\theta_a) - f_g(I_i;\theta_g)|_2 \tag{2.26}$$

Depending on the altitude from which an aerial picture was captured, the nearest object in the scene can be hundreds of meters away, or that the furthest object can be just a few meters away. In response to this ambiguity problem, the authors adapted the aerial image feature function $\mathrm{f_a}$ so that it considers images at multiple spatial scales.

The authors applied this training approach on two datasets: Charleston, introduced by Tian

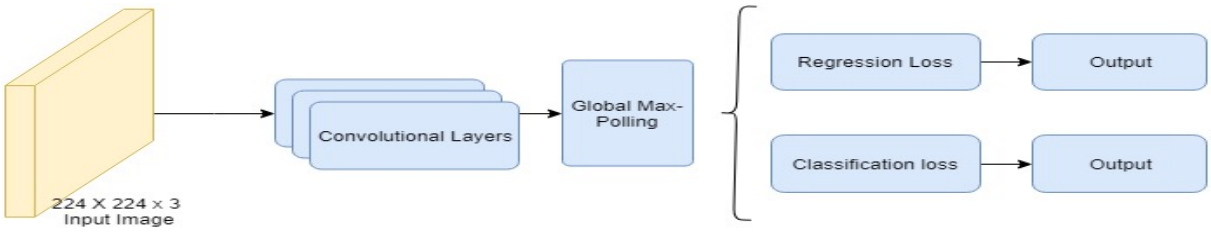


Figure 2.6: CNN architecture proposed by Vo et al. (2017).

et al. (2017), and San Francisco, introduced by Workman and Jacobs (2015). Using the Euclidean distance to localize a ground-level query image, they compared the ground-level feature $f_g(\mathrm{I};\theta_g)$ for the query image against a reference aerial image feature $f_a(l;\theta_a)$, at location $l$:

$$|f_a(l;\theta_a) - f_g(I;\theta_g)|_2 \tag{2.27}$$

The output can be the geolocation of the image that is the nearest neighbor of the query image in feature space, or it can be a list of candidate regions sorted by distance in feature space.

The authors also evaluated the localization performance obtained with off-the-shelf CNN features. Features were extracted from both the aerial and the ground-level query images using several network architectures trained for different target label spaces. There were two networks which outperformed all the remaining ones: GoogleNet (Szegedy et al. (2015)) and AlexNet (Krizhevsky et al. (2012)), both over the Places database (Zhou et al. (2014)).

#### 2.2.3.3 Revisiting IM2GPS in the Deep Learning Era

Vo et al. (2017) exploit both retrieval and classification approaches for image geocoding, proposing to combine deep feature learning as in PlaNet, introduced by Weyand et al. (2016), together with the Im2GPS approach, introduced by Hays and Efros (2008), in which a query image is matched against a database of geotagged images.

To perform a fair comparison, both introduced deep learning approaches used the same architecture, which is illustrated in Figure 2.6. The convolutional layers of the network were adapted from the VGG-16 network (Simonyan and Zisserman (2014)). Only the output layer and the loss function were changed, depending on the task. For classification, the authors appended a fully connected layer to the max pooling layer and used a softmax-crossentropy loss, while for the retrieval method they implemented the Distance Metric Learning (DML) loss.

The first approach consists of addressing geolocalization as a classification problem. The image's GPS coordinates were converted to classes, and each class corresponds a geographical

area on the map. Similarly to the adaptive scheme introduced in PlaNet, the authors repeatedly divided the Earth's surface, until the number of images in each cell, or the physical area of it, is smaller than a given threshold. In order to make a location prediction as precisely as possible, the authors studied 6 different partitions, generating 10, 80, 359, 1060, 1693 and 7011 regions, respectively. Having different partitions increases the amount of information preserved, since there are different proximity views of a single region. The authors also conducted an investigation using the 6 partitions simultaneously.

The second approach is based on geolocalization by image retrieval, in which the location of a query image is inferred by matching it to a dataset of geocoded images. To learn the representations for establishing the comparisons, the authors employed a distance metric learning approach, using pairs of images labeled as *similar* or *different*. In case those labels are not available, they can be automatically generated (Song et al. (2016)). The trained CNN can be used as a feature extractor, and at test time the model can either find the nearest neighbor of the query image in the feature space, or use a $k$-NN density estimation procedure.

The training data was retrieved from the Im2GPS dataset (Hays and Efros (2008)), and the test sets were built by the authors, making sure that no image in the training and test datasets came from the same photographer. The test set consisted of 3000 images from Im2GPS and 4000 images from the Flickr dataset (Thomee et al. (2016)). The training phase of the classification approach consisted of training two networks, denominated L and M respectively. The network L was trained with a classification loss corresponding to a partition of 7011 classes. The network M had 6 different losses since all the 6 partition schemes were used. Thus, this second network produces 6 localization outputs, each one of them being evaluated independently. From the training phase for the retrieval approach, it resulted a ranking network, denominated R. Additionally, the authors made experiments with different networks as the feature extractor: the aforementioned L and M networks, and the state-of-art models NetVLAD Arandjelovic et al. (2016) and Siamac Radenovic et al. (2016).

To evaluate the geolocalization accuracy, the authors measure the percentage of test images distributed over 5 error thresholds, corresponding to 5 levels of localization: street, city, region, country, and continent. The authors concluded that the geolocalization accuracy of the classification model output using 10 classes is quite bad. The authors also concluded that the geolocalization accuracy gets worse if the partitioning is too fine.

Regarding the retrieval model's performance, when considering localization by nearest neighbor image retrieval, all 5 models perform well, outperforming the classification result at the street

| | Street | City | Region | Country | Continent |
|---|---|---|---|---|---|
| Threshold (Km) | 1 | 25 | 200 | 750 | 2500 |
| Human assignment | | | 3.8 | 13.9 | 39.3 |
| Im2GPS (Hays and Efros (2008)) | | 12.0 | 15.0 | 23.0 | 47.0 |
| PlaNet (Weyand et al. (2016)) | 08.4 | 24.5 | 37.6 | 53.6 | 71.3 |
| Classification model with 7011 classes | 06.8 | 21.9 | 34.6 | 49.4 | 63.7 |
| Regression model | 12.2 | 33.3 | 44.3 | 57.4 | 71.3 |
| Regression model trained with 28M training photos | 14.4 | 33.3 | 47.7 | 61.6 | 73.4 |

Table 2.1: Accuracy results in the geocoding experiment reported by Vo et al. (2017).

and city levels. Using kNN-kernel density estimation improves the accuracy, and at the same scales makes the retrieval approach comparable with the classification approach.

### 2.2.4 Overview on the Related Work

Image classification is one of the fields in which the application of machine learning methods has most successfully contributed over the years. Among those methods, Section 2.2.1 reported six deep learning models. The first described method was a pioneering neural network, named LeNet, which is characterized by the low number of convolution layers and the inability of processing high-resolution images. As an attempt to address this issue, the AlexNet was introduced, It is a LeNet based model, but significantly deeper, which affects positively the accuracy results. The next introduced network was the VGG network, which outperforms the majority of the previously introduced networks, while being a considerable deep network with a high number of trainable parameters. Nonetheless, having deep and parameter overloaded networks increase the computation budget. The Inception was announced as being a deep network with great accuracy results while maintaining the complexity constant. Also under the same purpose, the MobileNet was introduced as being a small and efficient network, suitable for mobile applications. Lastly, Section 2.2.1 details the Xception network, a model entirely based on depthwise separable convolution layers.

This work also requires the application of image transformation methods, and so there were reviewed studies reporting two types of possible approaches to go for: image translation through unsupervised learning, and through style transfer. The described unsupervised learning image-to-image translation approach leverages cycle-consistent mapping functions. Regarding supervised learning methods, both Luan et al. (2017) and Chen and Schmidt (2016) addressed the task of style transfer, but the first aimed at solving image distortion and semantic variation problems, while the second introduced a patch-by-patch operation to swap the style.

As a result of applying deep learning methods for image classification, state-of-the-art methods for automatic image geocoding were introduced, some of them reviewed in Section 2.2.3. Vo et al. (2017) introduced a method, also described in that section, combining the PlaNet and the Im2GPS methods. Table 2.1 reports the results of the experiments performed by the authors on the Im2GPS test set. PlaNet obtains far more accurate results than Im2GPS, and it also outperforms the *Classification model with 7011 classes*. *Regression model* corresponds to the kNN kernel density estimation retrieval. The last row in the table refers to this same retrieval approach, but trained with a larger dataset (Yahoo Flickr Creative Commons 100 Million). This last model obtains the best results in all metrics, showing that the results improve when using a larger training set.

# 3 Automatic Geocoding of Historical Photos

This section details the approach used to address the task of automatically geocode historical photos using deep learning techniques. The method involves training two networks, a first one capable of transforming historical images and a second one aimed for geocoding those transformed images. Figure 3.1 illustrates an overview of the implemented solution, which consists of an end-to-end network combining two components. The application of the geocoding component leverages the ResNet50 network, and thus Section 3.1 describes the model. Section 3.2 details the coloring component, specifically its architecture and the methods used to make image transformations, whereas Section 3.3 describes the geocoding component. Finally, Section 3.4 overviews this entire chapter.

## 3.1 ResNet50 Network Structure

The residual network (ResNet) model (He et al. (2016)) is a CNN composed of residual blocks, and it was introduced with the purpose of solving the degradation problem. Over the years, with the evolution of computer processing capabilities, CNNs start becoming deeper. For instance, the LeNet network, presented by LeCun et al. (1998), was introduced with 7 layers, Simonyan and Zisserman (2014) pushed the depth of the VGG network up to 19 layers, and later Szegedy et al. (2015) introduced the Inception network which has more than 25 layers. Even though this depth increasing resulted in an improvement of the accuracy in image processing tasks (Simonyan and Zisserman (2014)), the authors of the ResNet model verified that with the increase of the network's depth, the accuracy starts getting saturated and degrading - the degradation problem. The graphics in Figure 3.2 are representative of this issue, as they establish a comparison between the training and test error of two networks trained on the CIFAR-10 set (Krizhevsky and Hinton (2009)). As illustrated, the deeper 56-layer network has a higher error in both the training and testing phases.

The ResNet models are based on residual leaning, which consists in, instead of having a stack of layers fitting the desired mapping $H(x)$, transforming the desired mapping into $F(x) = H(x) - x$, so that the networks learn a residual function $F(x)$, instead of learning a mapping

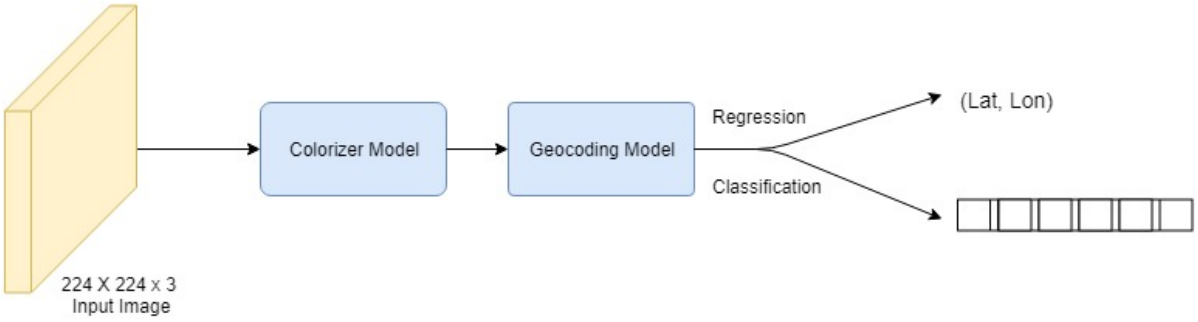


Figure 3.1: Overview of the introduced model.

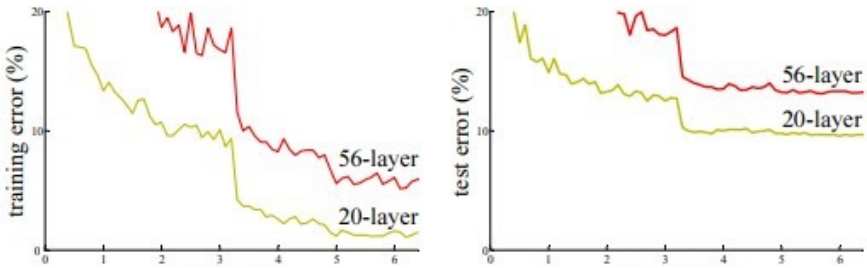


Figure 3.2: Image extracted from He et al. (2016). Left: Training Error. Right: Testing Error.

directly between $x$ and $H(x)$. This residual learning approach is applied to every residual building block, i.e., blocks that contain stacked layers, and are in charge of leading every set of layers to learn the residual function. A residual block is illustrated in 3.3. The picture contains a stack of layers learning the mapping $F(x)$, and a shortcut connection, also referred to as identity mapping, to add, element-wise, the input $x$ to $F(x)$. Notice that when a block is formed by convolution layers, this element-wise operation is performed channel by channel between two feature maps. The identity mapping does not introduce extra parameters, and in cases where $x$ and $F(x)$ have different dimensions, the connection can be used to perform projections so that both dimensions match.

In this work, I leveraged the 50-layer ResNet model (ResNet50), which architecture is illustrated in Figure 3.4. This network is based on residual blocks of three layers, instead of the two layers residual blocks used in RetNet34. Two of the three layers are $1 \times 1$ convolutions, and the middle one is a $3 \times 3$ convolution. The network is part of the applications of the high-level library Keras[1], and it is made available alongside pre-trained weights on ImageNet, which were later fine-tuned to fit the purpose of the tasks it was applied on.

Experiments conducted by the authors demonstrated the efficiency of using residual learning over plain networks that do not contain residual blocks. The authors took two plain networks

[1] `http://keras.io/applications/#xception`

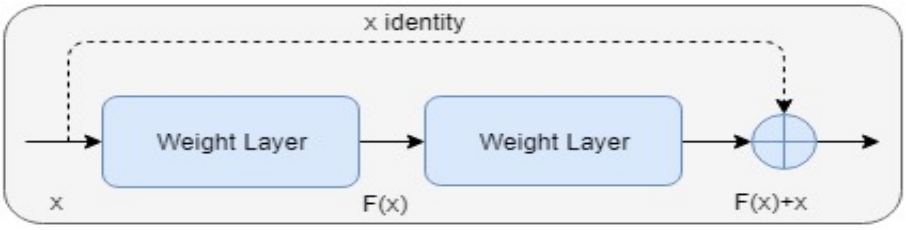


Figure 3.3: Residual learning: a building block.

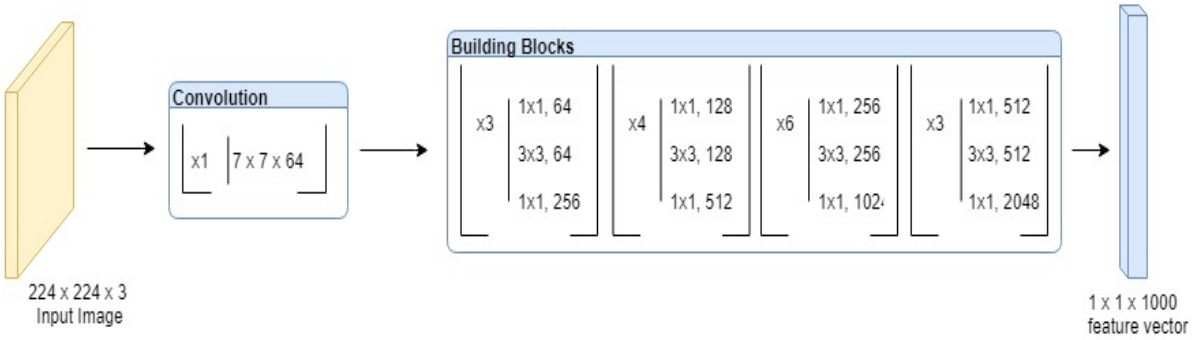


Figure 3.4: ResNet50 architecture.

with 18 and 34 layers respectively, and two other residual networks, also with 18 and 34 layers each. All four models were trained from scratch on 1.28 million training images, and evaluated on the ImageNet validation set (Russakovsky et al. (2015)). Both ResNet models performed better in comparison to the plain networks, with the ResNet with 34 layers performing better than all the other architectures, with a top-1 error of 25.03%. In another case study, I trained both the ResNet50 and the deepest network made available in Keras, the InceptionResNetV2 (Szegedy et al. (2017)) model, over the Old San Francisco Project dataset, to perform classification and regressions tasks. The results are displayed in Table 3.1. The ResNet50 model proved to be more accurate and required less computational effort in comparison to InceptionResnetV2 network. The changes made in the architecture of the InceptionResnetV2 network during this experiment were the same as performed in the ResNet50 architecture, which are described in Section 3.3. Moreover, an ensemble of residual networks achieved 3.57% error in the ImageNet test set, allowing the authors to win the *1st* place on the ILSVRC 2015 classification task[2]. The efficiency and the results of the experiments supported the decision of leveraging the ResNet50.

## 3.2 Coloring Historical Photos

Based on recent deep learning algorithms, a network was trained to perform image transformations, specifically transform grayscale images into colorful ones. As mentioned before, this is

[2]`http://image-net.org/challenges/LSVRC/2015/`

| | Accuracy | | | | Distance (Km) | |
|---|---|---|---|---|---|---|
| | <100m | <500m | <1Km | <5Km | Mean | Median |
| ResNet50 Regression | 0.098 | 1.788 | 6.878 | 78.514 | 3.829 | 2.941 |
| ResNet50 Classification | 0.460 | 5.334 | 15.987 | 68.487 | 4.307 | 3.101 |
| InceptionResNetV2 Regression | 0.098 | 1.834 | 7.423 | 76.914 | 3.917 | 3.046 |
| InceptionResNetV2 Classification | 0.287 | 3.501 | 10.758 | 62.581 | 4.793 | 3.776 |

Table 3.1: Experiment results using the ResNet50 and the InceptionResNetV2 models.

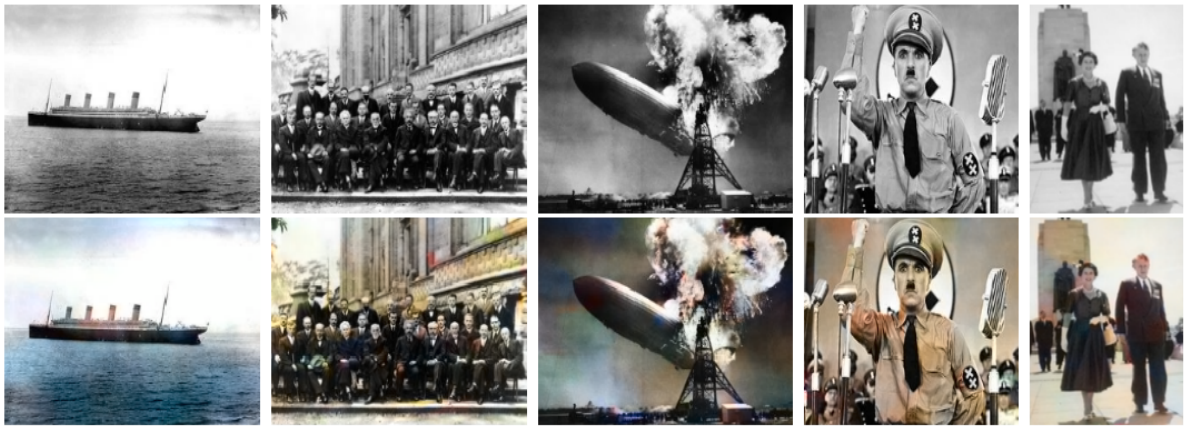

Figure 3.5: Examples of historical pictures colored using Baldassarre's model, made available by Federico Baldassarre (2017).

an important component for the resolution of the problem addressed in this work, as the geocoding network, which is introduced in Section 3.3, was pre-trained with modern colorful images. Thus, given a historical query photo, most likely in grayscale, the task is to color it so that the geocoding model can leverage the color information and perform content-based geolocation predictions.

The chosen network architecture to perform the task was made available by Majumdar in his GitHub repository[3]. It is fully based on a piece of work introduced by Federico Baldassarre (2017), in which the authors combined a deep CNN trained from scratch with the pre-trained high-level features extractor Inception-Resnet-v2. The network succeeds in coloring high-level image components such as the sky, sea, and forests. However, when it comes to reduced size concepts present in the pictures, the model is not that successful. Figure 3.5 shows the results of testing this Baldassarre's model on historical pictures.

The network's architecture is illustrated in Figure 3.6, and it is logically divided into the following four components:

[3] `http://github.com/titu1994/keras-mobile-colorizer`

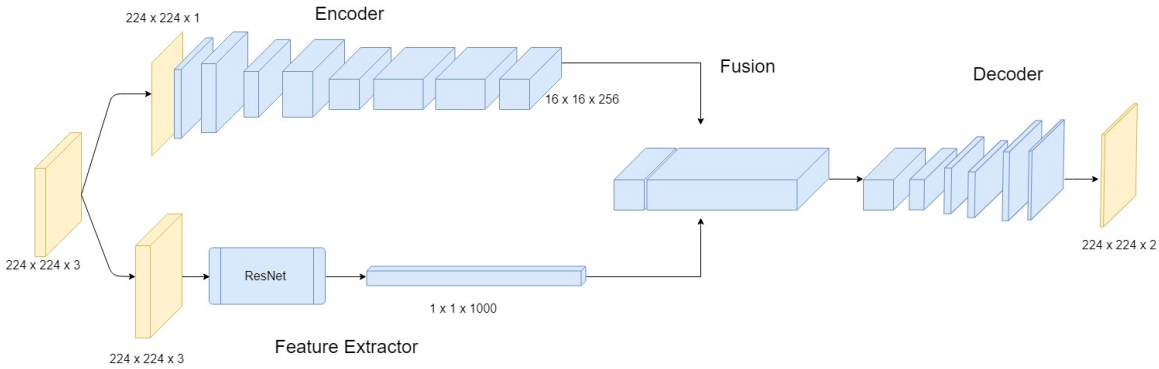


Figure 3.6: Coloring model architecture.

- The encoder component, which consists of a set of 8 convolutional layers. It processes $224 \times 224 \times 1$ images (i.e., only the luminance channel of the input), outputting a $16 \times 16 \times 256$ feature representation. Each convolution is followed by the ReLU activation function.

- The feature extractor component, which corresponds to the ResNet50 network. It processes $224 \times 244 \times 3$ images, and it returns the output of the layer immediately before the softmax function, which corresponds to a $1 \times 1 \times 1000$ feature tensor. This network was initialized with the ImageNet pre-trained weights available on Keras.

- The fusion component, which is responsible for concatenating both the output of the encoder with the output of the feature extractor, and is followed by another convolution layer, resulting in a new feature volume of $16 \times 16 \times 256$.

- The decoder component, which gathers the last set of layers, performs several convolutional and upsampling operations to the fusion component output, to obtain a final volume output of $224 \times 224 \times 2$ (i.e., the inferred color channel's values).

Given a grayscale input image in the CIEL*a*b* color space, the model processes the luminance component $X_L \in \mathbb{R}^{224 \times 224 \times 1}$ of an input image of dimensions $224 \times 224 \times 3$, and estimates the a*b* components, which correspond to the green-red and blue-yellow color values. As illustrated in Figure 3.6, the encoder only processes images with the luminance channel, while the feature extractor considers all three components. The output is a representation of the input image with two dimensions, which correspond to the coloring components.

As previously mentioned, the feature extractor was the only component with initialized weights. Given so, the coloring network was trained over 70000 randomly chosen modern pictures from the Yahoo Flickr Creative Commons 100 Million dataset. The entire network, including the parameters of the feature extractor, were trained using Adam optimizer with a learning rate

starting at 0.0001. To calculate the error and propagate it via back-propagation it was employed the mean square error between the estimated a*b* values and the ground-truth values as the loss function.

Notice that, when joining the coloring network with the network trained to geocode historical images, the input of the latter will be the result of the concatenation of the two dimensions output of the coloring model with the luminance channel of the query image. Summarizing:

- Given a query old photo, the luminance channel and the color channels of it are separated.
- The model predicts the color values using the method already described.
- The predicted color values are joint again with the luminance component, forming a representation of shape $224 \times 224 \times 3$ to be fed into the geocoding model.

## 3.3 Geocoding Historical Photos

In Section 2.2.3 several approaches are introduced for automatic image geocoding, which have shown to be effective in tackling this task both as a regression and a classification problem. Leveraging the ResNet50 network alongside the Keras deep learning library, and applying similar methods to previous solutions, this work introduces three different network typologies for three different approaches: the regression approach, in which the model predicts a pair of continuous variables corresponding to the latitude and longitude values; the classification approach, where the output variable takes class labels and represents a probabilistic distribution over them; and the regression + classification approach, which combines both regression and classification loss functions, outputting both types of output variables.

As the image transformation network is applied to old historical photos, the geocoding model can be pre-trained over modern data. As aforementioned, using the coloring component is highly important, since this way the geocoding models were trained over much larger datasets, which reflects on the accuracy of the predictions, particularly in regression tasks, as studies have shown (Vo et al. (2017)). So far, it was introduced an overview of the entire solution, which is illustrated by Figure 3.1, the way the query images are processed by coloring model, and how this last one feeds the geocoding model (see Section 3.2). Notice now to the implementation of the geocoding component.

Regarding the architecture of the model to perform the regression geocoding tasks, the network was configured by removing the last fully-connected layer from the original ResNet50's

architecture, and replacing it with a dense layer with an output dimensionality of two, which corresponds to the (latitude, longitude) pair of geospatial coordinates. At training time, the model learns to predict the location of a given query photo by redefining the network's weights through the back-propagation of the error. This error is calculated using the Vincenty's formulae, which were introduced by Vincenty (1975). The formulae correspond to a set of two iterative solutions used to calculate the distance between two points on the surface of Earth, assuming the planet has the shape of an oblate spheroid. The first formula, referred to as the direct method, computes the location of a point that is in a given distance and azimuth from another point. The second method, referred to as the inverse method, computes the geographical distance and the azimuth between two given points. Experiments conducted and reported by the author have proven that on the Earth ellipsoid one can achieve values accurate to within 0.5 millimeters when using the inverse method. In these experiments, the loss function corresponded to a special case of the inverse method of Vincenty formulae. The formula assumes an ellipsoid with equal major and minor axes, and it is defined as follows:

$$\arctan \frac{\sqrt{(\cos\phi_2 \cdot \sin(\Delta\lambda))^2 + (\cos\phi_1 \cdot \sin\phi_2 - \sin\phi_1 \cdot \cos\phi_2 \cdot \cos(\Delta\lambda))^2}}{\sin\phi_1 \cdot \sin\phi_2 + \cos\phi_1 \cdot \cos\phi_2 \cdot \cos(\Delta\lambda)} \tag{3.1}$$

In the previous equation, $\phi_1$ and $\phi_2$ correspond to the geographical latitude values in radians, and $\Delta\lambda$ represent the absolute difference between the geographic longitude values, which is also in radians.

The activation function assigned to the dense layer was the sigmoid function, which is constrained by horizontal asymptotes - i.e., horizontal lines from which the distance to the function's curve approaches to 0 -, as so each pair of geospatial coordinates of each training data instance were normalized to values defined in the [0, 1] range. The learning rate value was initially defined to 0.0001 and the optimizer used was Adam.

Focusing now on the classification approach, the ResNet50 architecture was once again adapted by removing the last fully-connected layer, replacing it with a dense layer with an output dimensionality equal to the number of target classes. During training time, the network was initialized with ImageNet weights, using Adam optimizer, an initial learning rate of 0.0001, and categorical crossentropy as loss function. At inference time, for each input image, the network builds a tensor representing the probabilistic distribution over the target classes. Those target classes correspond to certain regions of the planet, obtained from the subdivision of the

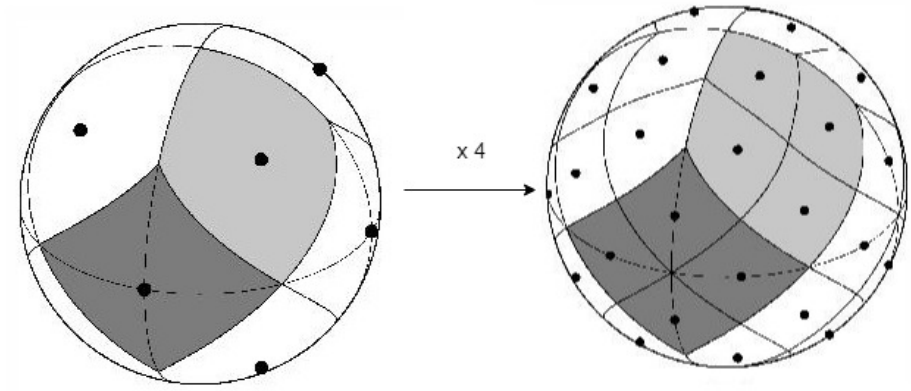


Figure 3.7: Application of the HEALPix algorithm.

Earth's surface into cells. To do so, it was leveraged the Healpy[4] Python wrapper for Hierarchical Equal Area isoLatitude Pixelation (HEALPix)[5], introduced by Gorski et al. (1999).

HEALPix is an algorithm used to project spherical objects into two dimensions space representations and perform points distributions as uniformly as possible over the projected surface (pixelization). Contrarily to the approach applied by Weyand et al. (2016) to divide the Earth's surface, in this algorithm each pixel covers the same surface area as every other pixel, as an attempt to keep every point equidistance from each other. Given a spherical surface, the partition of it proceeds by hierarchically tessellated it into curvilinear quadrilaterals. The lowest partition corresponds to 12 pixels, and the resolution of the tesselation increases by dividing each pixel into four new ones. Figure 3.7, which was extracted from the HEALPix website[6], illustrates the application of HEALPix, in which the resolution increases from 12 to 48 pixels. Each map is a numpy array, and each element corresponds to a location, and so this representation allows to map different Earth's regions to target classes.

The last introduced model corresponds to a neural network capable of performing both regression and classification tasks, i.e., it combines both types of outputs. The motivation to build and train a network in these terms had to do with the belief that using two loss functions could result in a more precise learning of the model parameters, which could lead to better accuracy results. The network was trained over the same regression and the classification losses described above. The procedures to build the target classes were also similar to the classification approach. Once again, the last fully-connected layer was removed from the original ResNet50's architecture, and it was replaced by set of dense layers, each set corresponding to the

[4] `http://github.com/healpy/healpy`
[5] `http://healpix.sourceforge.net`
[6] `http://healpix.sourceforge.io`

regression components and the classification components. These components follow the same aforementioned architecture, and during training there were used the same loss functions, initial learning rate value, and optimizers.

## 3.4 Overview

This chapter detailed the approach to automatically geocode historical photos. The solution consists on training two sets of networks to process images, both addressing different purposes, i.e., coloring images and geocode them. Both networks are joined together building an end-to-end system. The CNNs are entirely dependent on the ResNet50 network, which is introduced in Section 3.1.

As the task was divided into two subtasks, Section 3.2 introduced the approach to address the first subtask, i.e., coloring historical photos. It details the architecture of the proposed neural network model, which is based on an earlier work that is briefly introduced in this segment.

The last section, Section 3.3, reported the approach used to geocode historical photos, specifically the networks to perform geocoding as a regression task and a classification task. There were described the model's architectures, the way they were trained, and the method to perform a subdivision of the Earth's surface, in consideration to the classification task. It was also introduced a network that leverages both regression and classification loss functions, as an attempt to update the layer's weight values in a more precise way.

# 4 Experimental Evaluation

This chapter details the experimental evaluations conducted with the networks introduced in Section 3, and reports the corresponding results. Section 4.1 presents the datasets used to train and evaluate these networks, along with the evaluation methodologies. Next, Section 4.2 reports the performed experiments regarding the coloring component, and Section 4.3 presents and discusses the results of the experiments conducted on the proposed geocoding model, which integrates the coloring component. Finally, Section 4.4 gives an overview of the obtained results.

## 4.1 Datasets and Evaluation Methodology

As previously mentioned, with the emergence of the Web 2.0, one can have immediate and easy access to shared multimedia content, such as high-quality modern photos. Despite the extensive data offer, ready to be leveraged as the training dataset, these photos must be already accurately geocoded, so that the geocoding networks learn to perform the task with high precision. The dataset used to train the geocoding and the coloring networks with modern photos was the Yahoo Flickr Creative Commons 100 Million dataset. This dataset is the largest public multimedia collection available, containing 92.8 million photos, from which 48 million are geographically annotated. Table 4.1 describes the percentage of photos assigned to some visual concepts, which emphasizes the photos diversity of the dataset.

Instead of using the entire Flickr dataset, two bounding boxes were defined, from which resulted two new subsets containing photos from the cities of New York and San Francisco, respectively. Even though previous work has shown that the efficiency of neural networks increases with the growth of the training dataset size (Vo et al. (2017)), learning to geocode photos of a particular area can produce better accuracy results, whereas training a network to geocode photos worldwide may require more time and more computational effort. Moreover, New York and San Francisco are two of the few cities from where there are more than 10,000 unique users captured media available in this database. Additional information about the dataset and the bounding boxes is described in Table 4.2, while Figure 4.1 illustrates the heatmaps concerning the distribution of the photos in the collections of New York and San Francisco.

| Concepts | Percentage of photos |
|---|---|
| Outdoor | 44.291% |
| Indoor | 14.014% |
| Nature | 9.906% |
| Landscape | 6.063% |
| Building | 4.175% |
| Black and White | 2.585% |

Table 4.1: Percentage of photos per concept in the Flickr dataset

| | New York | San Francisco |
|---|---|---|
| Number of Flickr photos | 1139379 | 865977 |
| Number of historical photos | 37618 | 13255 |
| Area of bounding box( $km^2$) | 2303.27 | 2525.01 |
| Number of HEALPix cells (Flickr) | 820 | 421 |
| Number of HEALPix cells (Historical) | 328 | 63 |

Table 4.2: Statistical characterization of the evaluation datasets.

Regarding the datasets containing old photos, the experiments on the geocoding networks reported bellow used old photos extracted from the Old San Francisco Project and the Old New York Project. Both these projects consist of having web platforms where users can manually geocode historical photos. The photos came from the San Francisco Public Library's San Francisco Historical Photograph Collection and the New York Public Library's Milstein Collection, respectively. In the experiments, I only considered the geo-referenced data, which for the San Francisco collection was done both manually and using the Google Geocoding API[1]. Table 4.2 shows additional information about the datasets, while Figure 4.2 illustrates some examples of photos from each dataset. It is important to notice that the majority of the photos have bad quality, low resolution, and noise, which may hamper content-based geocoding.

In order to appraise the geocoding results, I used an evaluation methodology similar to the one introduced in the work from Hays and Efros (2015), which was then applied in several studies related to this work (Weyand et al. (2016), Vo et al. (2017)). In fact, the authors conducted some experiments which aimed to the task of geocoding, leveraging state-of-the-art algorithms, and evaluated the results through approximation scales of streets, cities, regions, countries, and continents. In the experiments in this work, I followed on the same idea, adapting the scales. Considering the coverage area of the geocoded photos in the datasets used in these studies, there were quantified the fraction of images localized within the following given radius: 100

[1] `http://developers.google.com/maps/documentation/geocoding/intro`

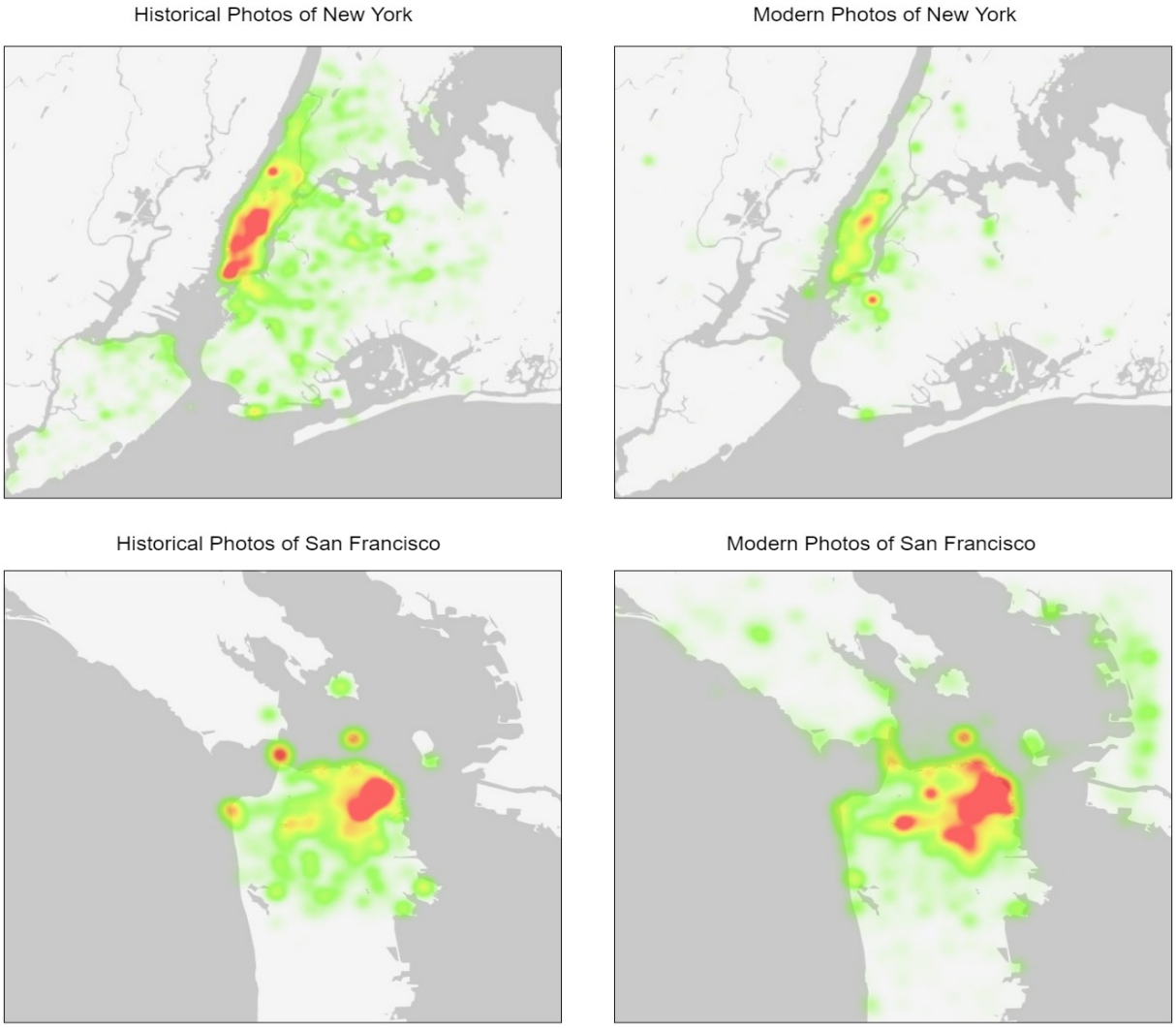


Figure 4.1: Density of photos in each of the evaluation dataset.

meters, 500 meters, 1 kilometer, and 5 kilometers from the ground truth position of the query. Additionally, mean and median error distance were measured for each experiment.

During the training and the evaluation of the geocoding models described in Section 4.3, the datasets were divided as follows. Considering first the Flickr dataset, 80% of the data was used to train the models, and 20% to perform the evaluations; regarding the historical datasets, due to the low number of instances, the experiments were performed by cross-validations, i.e., the datasets were divided in two equal parts, and each model was trained twice, so that each part obtained from the division could be used for both training and validation. To train the coloring network, 70000 pictures were randomly selected from the Flickr dataset.

Overall, the chosen datasets to train and evaluate the networks are suitable, as every instance is accurately geocoded. Concerning the Flickr dataset, the photos illustrate a wide variety of concepts, and so the coloring component will learn to map grayscale images to color values from

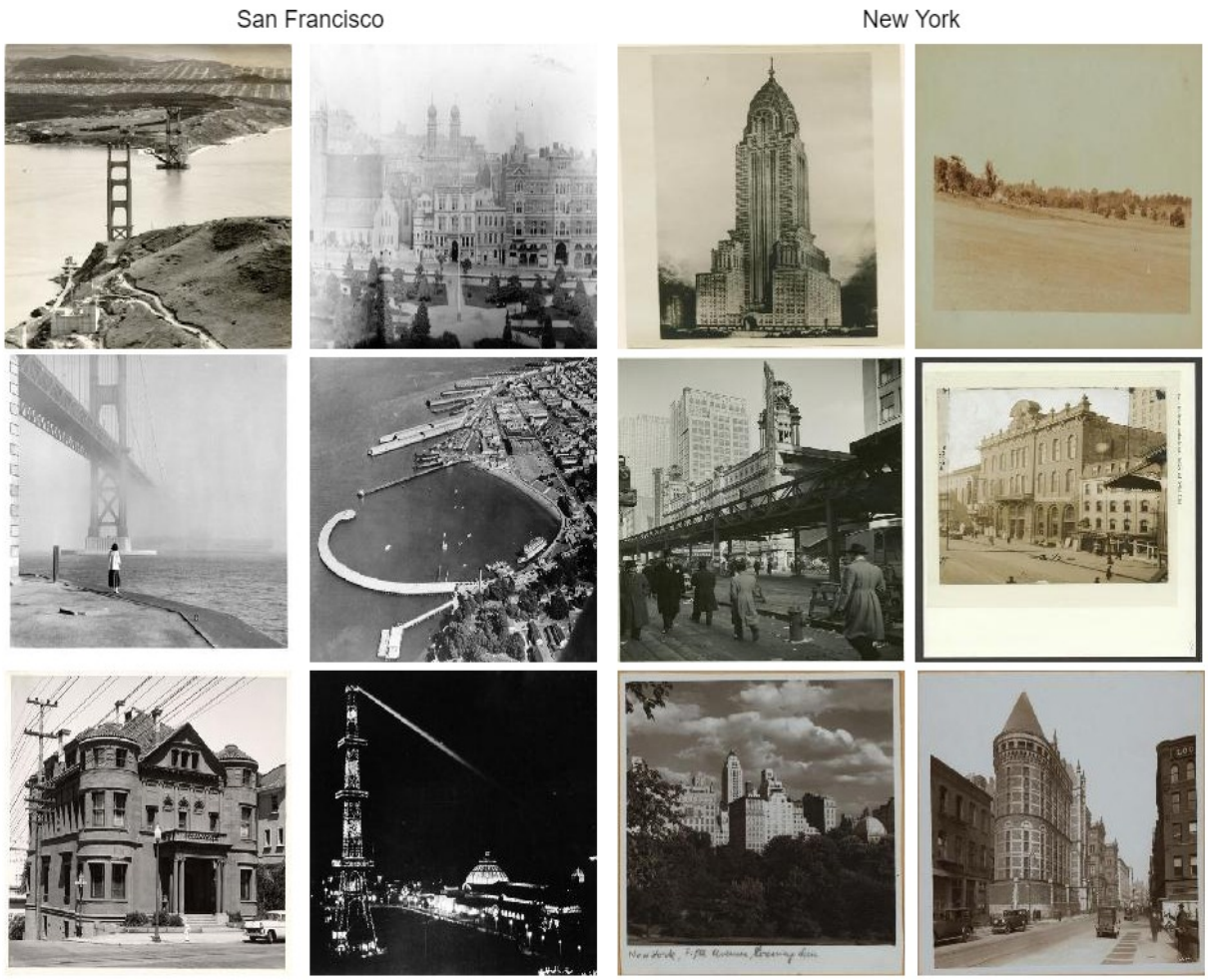


Figure 4.2: Examples of instances from each testing dataset.

a set of photos illustrating a significant variety of objects and landscapes, with a wide range of color values. The heatmaps show that the modern photos cover a big part of the cities, allowing the geocoding models to learn as many different points of the cities as possible. In the city of New York, the pictures are equally distributed, while in San Francisco there is a big concentration of photos in a single area. Regarding the datasets containing old photos, the heatmaps show that they are also covering a big area of the cities. In both cities, there is a single area with a big concentration of photos, which are equally significant. Some instances do not have as much quality as it would be desirable, however, they are all geocoded, which was a major factor when choosing these two historical datasets to train and evaluate the models.

## 4.2 Experimental Results using the Coloring Model

The coloring model was trained and evaluated using first the MobileNet, and then the ResNet50 network, both as feature extractors. Majumdar, the author of the GitHub repository

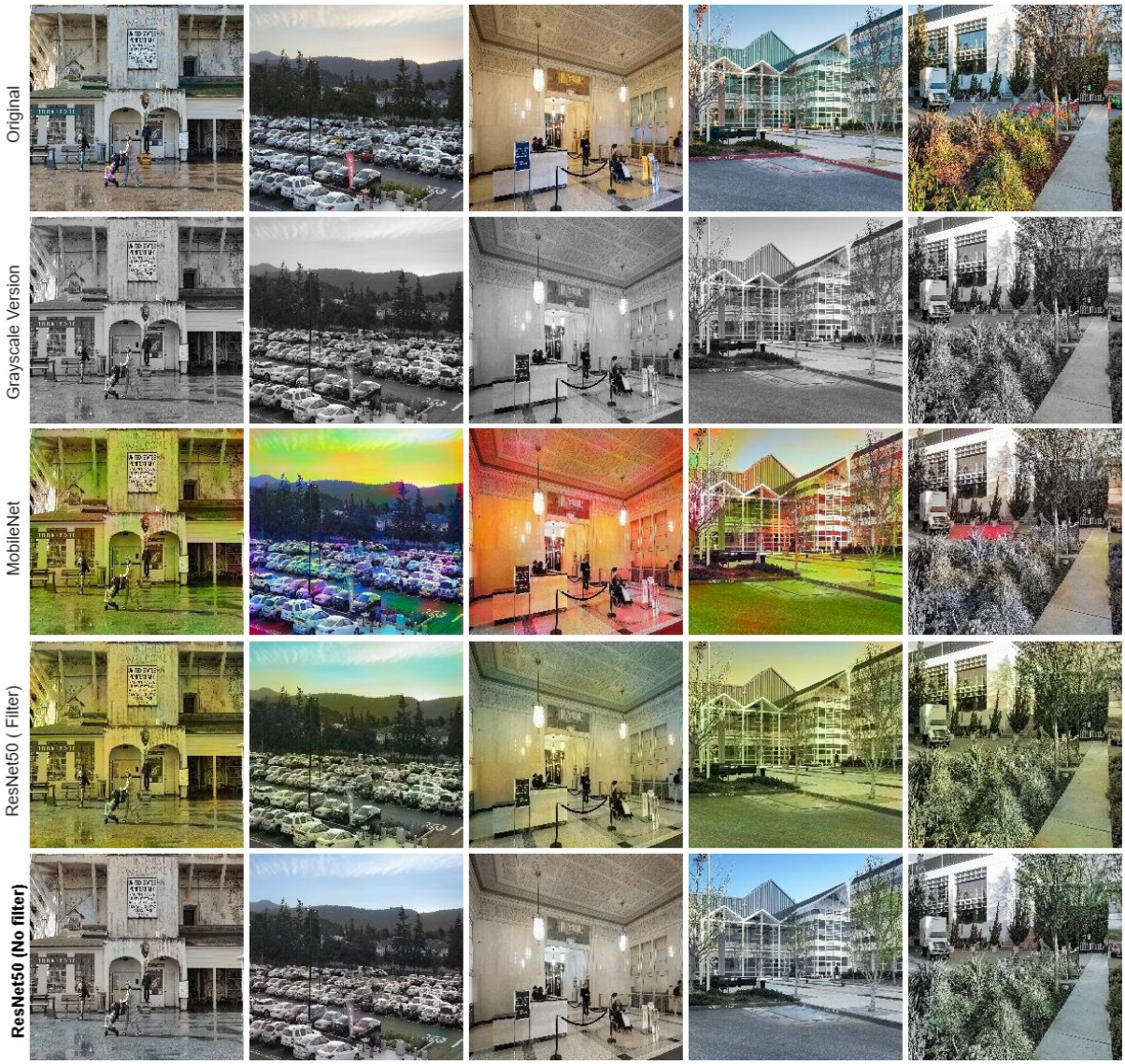


Figure 4.3: Results of applying the coloring model on the Guardian dataset.

that contains the architecture from where the coloring model was based on, reported his results using the MobileNet as feature extractor. Thus, a first experiment was conducted using that same feature extractor in order to build a baseline for further comparisons. Afterwards, a second experiment was conducted, by replacing the MobileNet with a ResNet50 architecture, which is the same model used in the geocoding task, as this network is much deeper, containing more weighted parameters to be trained and adapted to this coloring assignment. In both cases, the feature extractors were configured to output the feature vector corresponding to the output of the last layer before the fully-connected layer.

Figures 4.3 and 4.4 show example test results of applying the models. The model was evaluated on: i) pictures extracted from an article published in the Guardian journal[2], illustrating

[2]`https://www.theguardian.com/us-news/2016/feb/04/san-francisco-then-and-now-super-bowl-50`

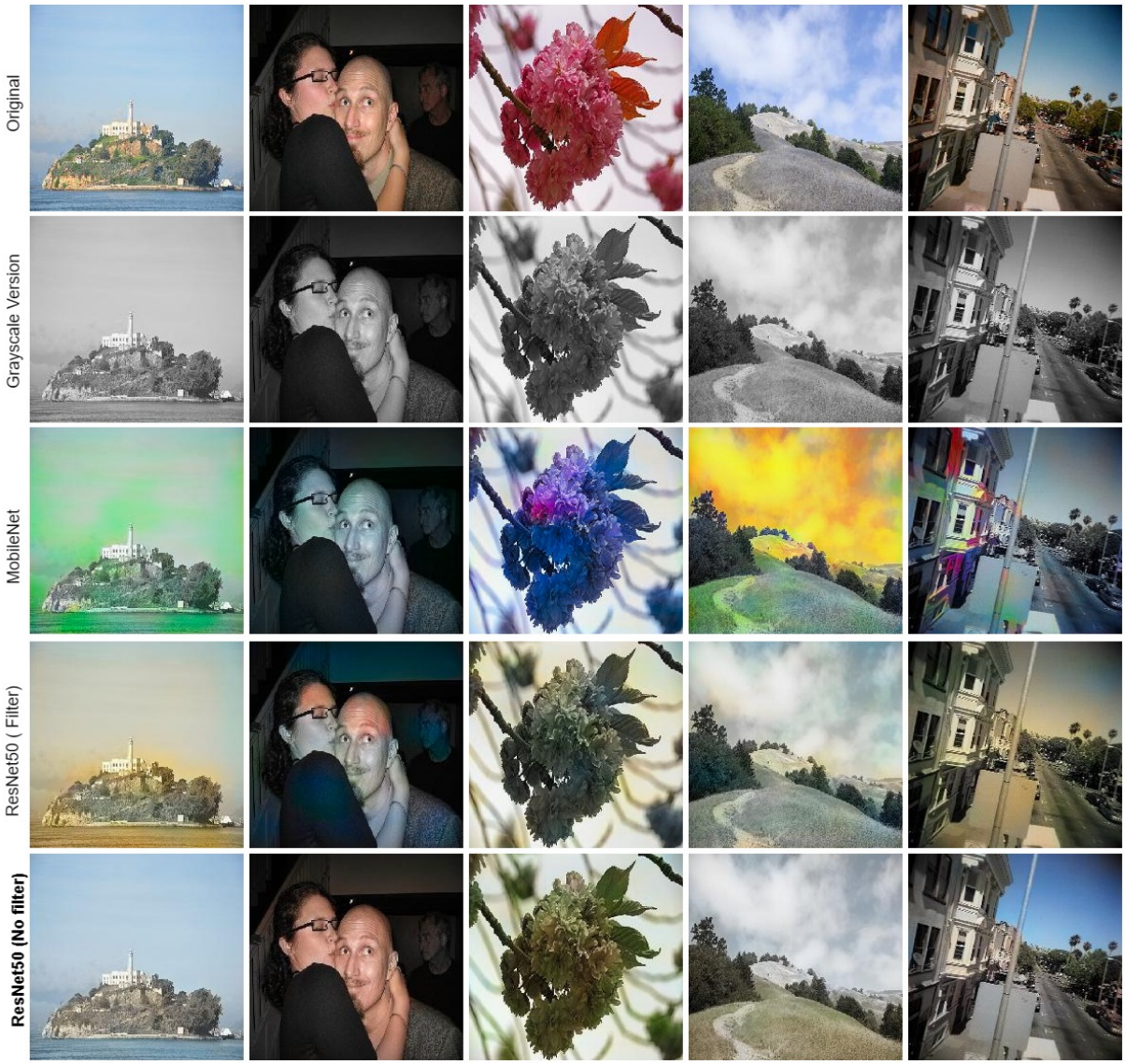


Figure 4.4: Results of applying the coloring model on the Flickr dataset.

the transformations of sites across the San Francisco Bay Area (Figure 4.3), ii) a subset from the model's training set (Figure 4.4), iii) and on the aforementioned sets gathering old photos. Given the colored modern photos ($1^{st}$ row in both figures), a function was employed to convert them into grayscale ($2^{nd}$ row in both figures), which were then processed by the coloring model. The third column of both pictures illustrates the results of using the MobileNet as feature extractor, whereas the last column regards the application of the ResNet50 as the feature extractor.

When using the MobileNet network as feature extractor ($3^{rd}$ row in both Figures 4.3 and 4.4) one can see that there are some images not entirely colored, remaining in a grayscale tone (e.g., $2^{nd}$ picture in Figure 4.4). On the other hand, there are photos entirely colored without the model making any semantic consideration, i.e., color values applied to pixels, regardless if they are, for instance, representing the sea or the grass (e.g., $1^{st}$ picture in Figure 4.3, $1^{st}$

| Sets of comparison | Average RMSE | |
|---|---|---|
| | Guardian dataset | Flickr dataset |
| Original $\longleftrightarrow$ Grayscale version | 18.608 | 22.687 |
| Original $\longleftrightarrow$ MobileNet | 47.683 | 30.925 |
| Original $\longleftrightarrow$ ResNet50+Filter | 34.329 | 30.210 |
| Original $\longleftrightarrow$ ResNet50 | **18.540** | **21.005** |

Table 4.3: Average RMSE obtained from evaluating the different coloring networks.

picture in Figure 4.4). Also, as a consequence of the low complexity of the MobileNet model, the architecture does not succeed in applying appropriate color values in the pixels, even when it can distinguish high-level features such as the sky and sea (e.g., $2^{nd}$ picture in Figure 4.3, $4^{th}$ picture in Figure 4.4).

Conversely, using a deeper feature extractor such as the ResNet50 ($5^{rd}$ row in both Figures 4.3 and 4.4) allowed the network to recognize and apply appropriate color values in high-level features such as the sky, sea, and the grass (e.g., $4^{th}$ picture in Figure 4.3, $4^{th}$ picture in Figure 4.4), resembling Federico Baldassarre (2017). In the case of the coloring component can not identify the features in an image, contrarily to what happens in the network using the MobileNet feature extractor, it does not apply wrong color values to the pixels, leaving the image almost identical to its original appearance (e.g., $3^{rd}$ picture in Figure 4.3).

In order to quantitatively evaluate the performance of the coloring component, similarly to Deshpande et al. (2015), I measured the average root mean square error (RMSE) between the original a*b* values of photos and the a*b* values of colored versions. As aforementioned, the a*b* values correspond to the color values of pictures represented in the CIEL*a*b* color space. The RMSE present a measure of the differences between the values predicted by a model and the actual values. In this specific case, I compared the pixel values between the original colored photos and the inferred ones. For $M$ images, the applied RMSE formula is defined as follows:

$$\frac{1}{M}\sqrt{\sum_{m_j=1,n_j=1}^{M_j,N_j}(y_{n_j}-y_{m_j})^2} \tag{4.1}$$

In the previous equation, $M_j$ and $N_j$ represent the pixels of the original and predicted photos respectively, and $y$ refers to the pixel value. Table 4.3 report the average RMSE values, from which one can conclude that the model using the ResNet50 outperforms the model using the MobileNet by a large margin. Again, these disparities in the results have to do with the different complexity of both feature extractor networks.

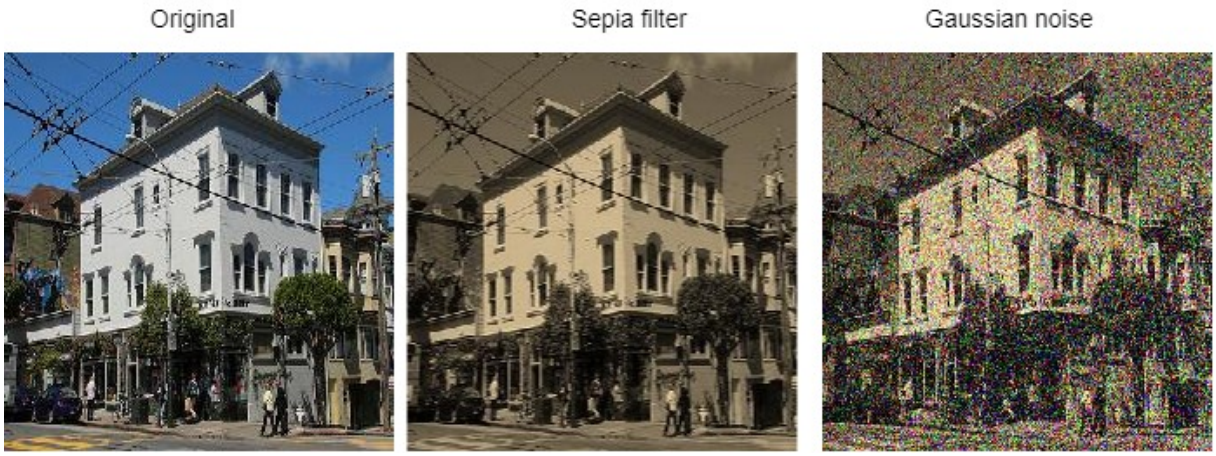


Figure 4.5: Pre-processing stages of a training image.

In the aftermath of going through the old photos in the datasets, the visual analysis of their appearance led to the conclusion that, besides the fact that they are in grayscale, some of them are also in a tone similar to sepia, and a few others with a really poor quality. These factors supported the need of doing a pre-processing of the photos, as an attempt to assign these same characteristics to the training data, The mentioned approach consists of two procedures:

- Apply a sepia-toning effect;
- Use a noise function to apply random variations of the brightness and color values. The leveraged function is based on one made available by Blanco-Silva[3], in which for each pixel of the image it is added a random value with normal distribution - Gaussian noise - between 0.001 and 0.026.

Figure 4.5 illustrates a training image with the sepia filter, as well as the noise function applied to it. Since it was already concluded that the model performs better when using the ResNet50 as feature extractor, in comparison to the one using the MobileNet architecture, a second case study was conducted. This time, the model using the ResNet50 as feature extractor was compared against the same model, using the same feature extractor, and the additional filters applied to the input training photos. In the following paragraphs and in the presented figures, this latter model is referred to as ResNet+Filter.

Some examples test results are displayed in the $4^{th}$ row of both Figures 4.3 and 4.4. Visually, the ResNet+Filter model produces better results, in comparison to the model which leverages the MobileNet architecture, in both the Guardian and the Flickr datasets. However, some pictures are still completely colored without any semantic differentiation (e.g., $4^{th}$ picture in

[3] `https://hub.packtpub.com/mathematical-imaging/`


Figure 4.6: Results of applying the coloring model on the old historical photos.

Figure 4.3). Moreover, since the model learned from images with the sepia filter, some results have that same tone (e.g., $1^{st}$ picture in Figure 4.4). When analyzing Table 4.3, the average RMSE values indicate that the ResNet50+Filter model is indeed more efficient than the one using MobileNet. Nevertheless, the model using the ResNet50 architecture outperforms by a large margin the ResNet50+Filter version. Given this results, one can say that there is no need to apply additional computational efforts in trying to filter the grayscale input photos since the model performs better without those.

In a last case study, all three models are evaluated with old photos from San Francisco and from New York. Examples results are shown in Figure 4.6. Since there are no original colored photos, it was not possible to perform quantitative evaluations. Visually, the results do not show any disparity from what was already concluded in the previous experiments, with the

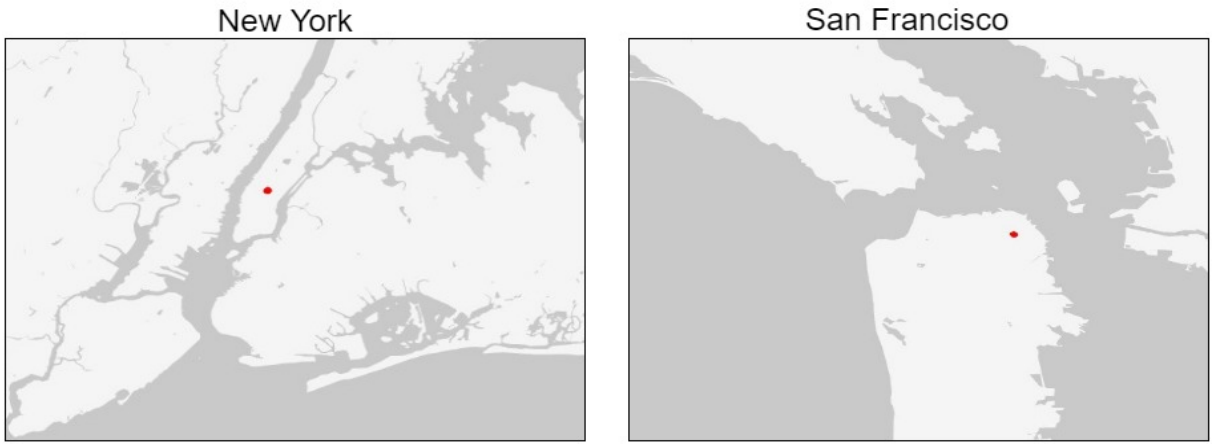


Figure 4.7: Centroid for the HEALPix cell with the highest density.

model using the MobileNet producing the worst results in comparison to the other two models. Some pictures are still entirely colored without any semantic consideration (e.g., $1^{st}$ and $3^{rd}$ pictures). The ResNet50+Filter model continues to produce results with some sepia tones (e.g., $3^{rd}$ and $5^{th}$ pictures). The model using the ResNet50 produces the best results.

Both qualitative and quantitative evaluations made in these experiments support the decision of leveraging the model using the ResNet50 as feature extractor, without applying any additional filters to the input training photos, besides converting them into grayscale, as the coloring component in this work.

## 4.3 Experiments Results using the Geocoding End-to-End Network

The experiments on the networks trained to perform geocoding tasks can be divided in two case studies: one concerning the city of San Francisco (Table 4.4), and the other targeting the city of New York (Table 4.5). Both tables report on the percentage of photos for which the estimated coordinates are within a given distance threshold of the ground-truth coordinates. The distances were measured using the Vincenty's geodetic formulae, which Section 3.3 presents and describes.

Both Table 4.4 and 4.5 are equally divided in two sets. This division is marked by the double line in the tables. The first set corresponds to the baselines and the experiments defined and performed using the modern data from the Flickr dataset, whereas in the second set it was used the historical dataset to define the baselines and perform the experiments.

The first set is divided as follows:

- The first group presents the baseline results. There are two types of baselines: the ones generated by random assignment of coordinates within a bounding box covering the photo's coordinates, and the ones resulting from the assignment of all photos to the centroid coordinates of the HEALPix cell with more photos in each dataset. The partition resolution of the Earth's surface was the same as used in the network's experiments. Additionally, there were defined two other *Most Frequent Cell* baselines, a finer-grained and a coarser-grained partition, in comparison to the actual partition used in the experiments.
- The second group reports the results of evaluating all three types of geocoding networks, i.e., regression, classification, and regression+classification (combined) networks, trained to geocode modern photos from the Flickr dataset;

The second set is divided as follows:

- The first group presents the baseline results. The baselines were defined in the same way as for the dataset of modern photos.
- The second group corresponds to the experiments on the same networks as in the second group of the first set, but this time trained and evaluated on historical data;
- The experiments conducted in the third group leveraged the introduced end-to-end network, which integrates both the coloring component and the geocoding component. The networks were trained and evaluated on historical data. Notice that in these experiments, the geocoding component was not pre-trained.
- In the fourth group of experiments, there were leveraged the geocoding networks from the first set, which were trained on modern photos. The three networks were evaluated on historical data. The fourth line of the group (*Flickr + Col. + Hist. R/C*) corresponds to evaluating the end-to-end network on historical data, using the pre-trained *Flickr Combined (R/C)* network from the first set, as the geocoding component.
- Finally, the last group reports the evaluation results from testing the same end-to-end network as in *Flickr + Col. + Hist. R/C*, but re-trained with historical photos.

Figure 4.7 contains two maps representing the HEALPix cells containing most photos in it. For this specific partitioning resolution, the most frequent cell is the same for both historical

| | Accuracy | | | | Distance (Km) | |
|---|---|---|---|---|---|---|
| | <100m | <500m | <1Km | <5Km | Mean | Median |
| Flickr Random | 0.002 | 0.025 | 0.117 | 3.043 | 31.584 | 28.803 |
| Flickr Most Frequent Cell (MFC) | 0.156 | **3.558** | 12.587 | 71.557 | **4.342** | 3.212 |
| Flickr MFC (fine-grained partition) | **1.077** | 8.003 | **16.756** | 70.125 | 4.456 | 3.192 |
| Flickr MFC (coarse-grained partition) | 0.005 | 0.646 | 4.237 | **77.070** | 4.357 | **3.057** |
| Flickr Regression (R) | 0.063 | 1.593 | 6.185 | 69.714 | 4.711 | 3.531 |
| Flickr Classification (C) | 0.134 | 2.221 | 8.219 | 67.183 | 4.895 | 3.590 |
| Flickr Combined (R/C) | 0.065 | 1.517 | 5.900 | 66.947 | 4.929 | 3.696 |
| Hist. Random | 0 | 0.045 | 0.151 | 2.958 | 31.204 | 28.263 |
| Hist. MFC | 0.913 | 6.511 | **24.738** | 74.772 | 3.656 | 2.218 |
| Hist. MFC (fine-grained partition) | **0.943** | **11.181** | 22.641 | 75.141 | 3.712 | 2.248 |
| Hist. MFC (coarse-grained partition) | 0.030 | 0.958 | 5.915 | 78.250 | 3.600 | 2.771 |
| Historical R | 0.098 | 1.788 | 6.878 | **78.514** | 3.829 | 2.941 |
| Historical C | 0.460 | 5.334 | 15.987 | 68.487 | 4.307 | 3.101 |
| Historical R/C | 0.166 | 3.282 | 10.977 | 73.030 | 4.284 | 2.861 |
| Coloring + Hist. R | 0.038 | 1,690 | 7.349 | 78.582 | 3.772 | 2.853 |
| Coloring + Hist. C | 0.325 | 4.142 | 13.052 | 65.613 | 4.595 | 3.519 |
| Coloring + Hist. R/C | 0.075 | 2.271 | 8.563 | 77.496 | 3.856 | 2.779 |
| Flickr + Historical R | 0.023 | 2.309 | 8.050 | 74.840 | 4.027 | 3.136 |
| Flickr + Historical C | 0.309 | 3.508 | 11.279 | 71.415 | 4.170 | 3.087 |
| Flickr + Historical R/C | 0.098 | 1.788 | 7.016 | 71.792 | 4.237 | 3.383 |
| Flickr + Col. + Hist. R/C | 0.189 | 6.805 | 18.657 | 73.776 | **3.536** | **2.114** |
| Flickr Pre-Training | 0.068 | 2.165 | 8.389 | 70.796 | 4.162 | 3.370 |

Table 4.4: Experimental results with photos from San Francisco.

and modern sets of both New York and San Francisco cities. Regarding the datasets of old photos from San Francisco and New York, the most frequent cell contains 23.810% and 9.596% of the total number of both training and evaluating photos, respectively. For the datasets of modern photos from San Francisco and New York, the most frequent cell contains 11.155% and 11.902% of the total number of training photos, and 11.153% and 11.950% of the total number of evaluating photos, respectively. Introducing these baselines allowed a comparison between results obtained from using simple heuristics, and from applying machine learning algorithms.

On what regards Table 4.4, specifically the results of evaluating the three types of geocoding networks on modern photos of San Francisco from the Flickr dataset, one can see that the regression network performed the best of the three typologies. When considering the networks trained and evaluated with historical data of the same city, the regression network was, once again, the best one, almost outperforming the *Hist. Most Frequent Cell* baseline. The networks trained with both regression and classification losses were expected to perform the best, since the trainable parameters of the layers were being updated considering two loss functions, however,

| | Accuracy | | | | Distance (Km) | |
|---|---|---|---|---|---|---|
| | <100m | <500m | <1Km | <5Km | Mean | Median |
| Flickr Random | 0.004 | 0.037 | 0.139 | 3.396 | 19.789 | 19.739 |
| Flickr Most Frequent Cell (MFC) | 0.050 | 5.697 | **12.836** | 53.393 | **6.075** | **4.493** |
| Flickr MFC (fine-grained partition) | 0.007 | **6.332** | 7.072 | 30.517 | 8.165 | 7.645 |
| Flickr MFC (coarse-grained partition) | 0 | 0.428 | 0.949 | 44.619 | 7.118 | 5.608 |
| Flickr Regression (R) | **0.078** | 1.764 | 6.250 | **53.786** | 6.255 | 4.613 |
| Flickr Classification (C) | 0.030 | 1.967 | 5.971 | 44.582 | 7.288 | 5.592 |
| Flickr Combined (R/C) | 0.069 | 1.595 | 5.689 | 52.489 | 6.395 | 4.748 |
| Hist. Random | 0.003 | 0.037 | 0.149 | 3.390 | 20.217 | 19.955 |
| Hist. MFC | 0 | **3.825** | **10.460** | 45.449 | **7.161** | 5.493 |
| Hist. MFC (fine-grained partition) | **0.080** | 2.012 | 3.015 | 22.345 | 9.852 | 8.476 |
| Hist. MFC (coarse-grained partition) | 0 | 0 | 0.872 | 38.397 | 8.269 | 6.287 |
| Historical R | 0.064 | 1.042 | 3.857 | 46.515 | 7.274 | 5.391 |
| Historical C | 0.024 | 1.327 | 4.418 | 34.130 | 9.082 | 7.536 |
| Historical R/C | 0.046 | 1.133 | 4.447 | 46.425 | 7.355 | 5.475 |
| Coloring + Hist. R | 0.016 | 0.588 | 2.430 | 42.161 | 7.585 | 5.817 |
| Coloring + Hist. C | 0.013 | 0.909 | 2.930 | 29.736 | 10.023 | 8.559 |
| Coloring + Hist. R/C | 0.032 | 0.753 | 2.773 | 43.612 | 7.535 | 5.719 |
| Flickr + Historical R | 0.066 | 1.706 | 5.760 | 45.192 | 7.486 | 5.693 |
| Flickr + Historical C | 0.035 | 1.446 | 4.732 | 38.979 | 8.209 | 6.542 |
| Flickr + Historical R/C | 0.077 | 1.536 | 5.561 | 44.649 | 7.503 | 5.785 |
| Flickr + Col. + Hist. R/C | 0.016 | 0.718 | 2.999 | **47.347** | 7.451 | 5.409 |
| Flickr Pre-Training | 0.006 | 0.423 | 2.199 | 47.158 | 7.306 | **5.340** |

Table 4.5: Experimental results with photos from New York.

that was not the case, neither with modern nor historical photos. One possible explanation has to do with the fact that a significant percentage of the training photos are assigned to a unique HEALPix cell, in both old and modern datasets.

Figure 4.8 shows two heatmaps illustrating the predicted coordinates distribution for both old and modern datasets for the city of San Francisco, using the regression networks. In the network trained with modern data, it is noticeable that the result's concentration covers the coastline, where the majority of the photos are truly assigned. Regarding the *Flickr Combined (R/C)* network, the concentration of the coordinates predicted moved from the coastline towards the region where the HEALPix cell containing more photos is located. Also, the set of coordinates inferred by classification network were entirely assigned to the cell containing most photos, and to the surrounding cells. In the experiments evaluating the regression network on historical data, the results distribution covers a big part of the coastline as well as the inner region of the city, resembling the distribution of the results from the *Historical R/C* network. The classification network, once again, inferred locations in the most frequent cell, and the areas around. These

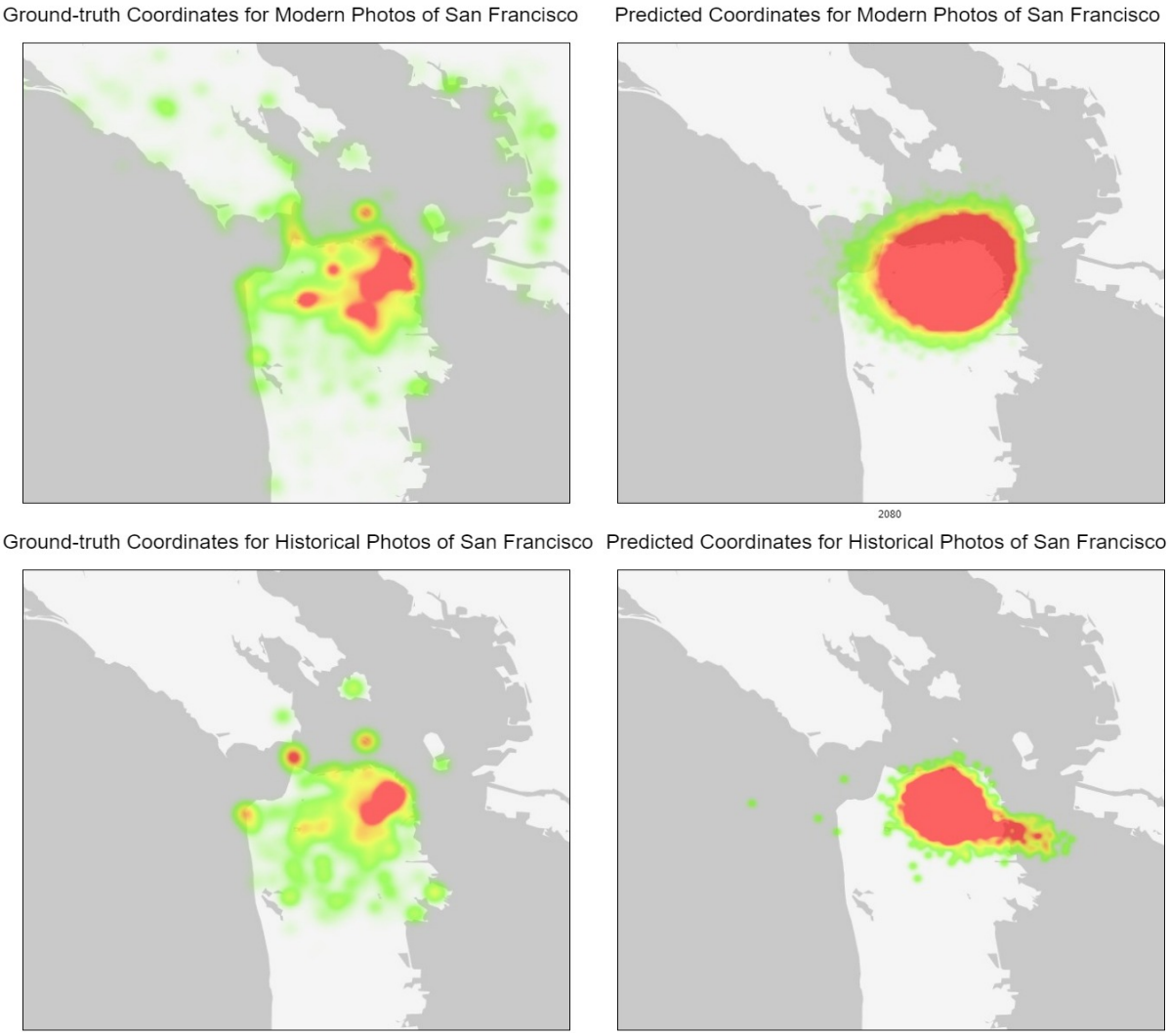


Figure 4.8: Density maps with basis on the ground-truth coordinates of the historical and modern photos from San Francisco, versus the estimated coordinates by the neural networks using a regression loss.

analyses of the result's heatmaps highlight the impact of having a big number of photos in the same cell, or near it, mainly in the networks trained with both regression and classification losses.

Evaluation historical photos without coloring them on networks trained with modern data ($4^{th}$ group from the $2^{nd}$ set) had an end goal of getting to know about the impact of having a coloring component in the introduced end-to-end geocoding model, and to verify if the results are in fact better when using the additional coloring component. Thus, the results of these experiments were compared with the results of evaluating the same historical data on the introduced end-to-end network (last experiment of the $4^{th}$ group, and $6^{th}$ group, both from the $2^{nd}$ set). The results of the experiments show that using the coloring component indeed improves the results, since the end-to-end network performed better, which highlights the importance of

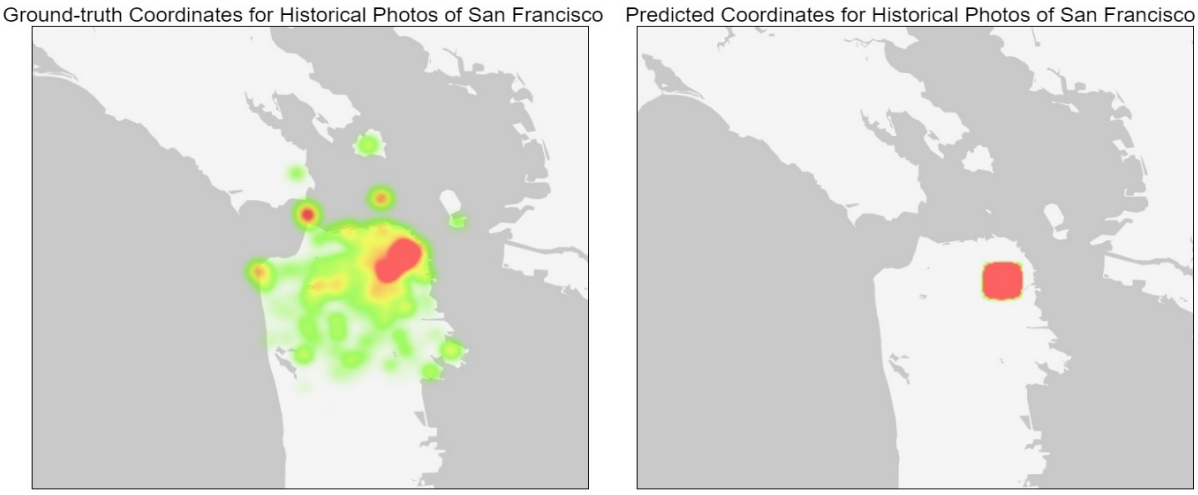


Figure 4.9: Density maps with basis on the ground-truth coordinates of the historical photos from San Francisco, versus the estimated coordinates by a end-to-end network.

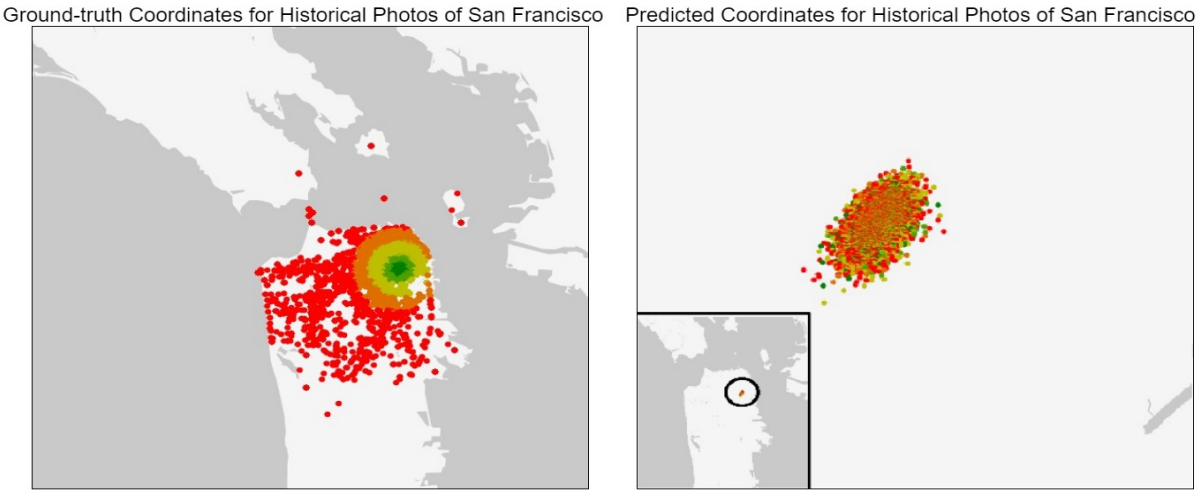


Figure 4.10: Error distribution with basis on the ground-truth coordinates of the historical photos from San Francisco, and the estimated coordinates by a neural network combining regression and classification losses.

having a coloring network capable of producing good results.

Focusing now in the introduced end-to-end network, results show that using a geocoding component pre-trained with modern photos and combined with the coloring network produces better results in comparison to re-training the same end-to-end network with historical data. From analyzing the heatmap with the distribution of the coordinates predicted by the best end-to-end trained network, i.e, *Flickr + Col. + Hist. R/C*, illustrated in Figure 4.9, it is noticeable that the coordinates are concentrated in the same area as the HEALPix cell containing more photos of San Francisco. Even though the error is lower when the end-to-end network is not fine-tuned with historical data, I argue that these results have to do with the fact of the predicted geospatial coordinates were near an area with a high photos concentration, which resulted in a lower distance error, whereas the other end-to-end network learned to infer coordinates in a

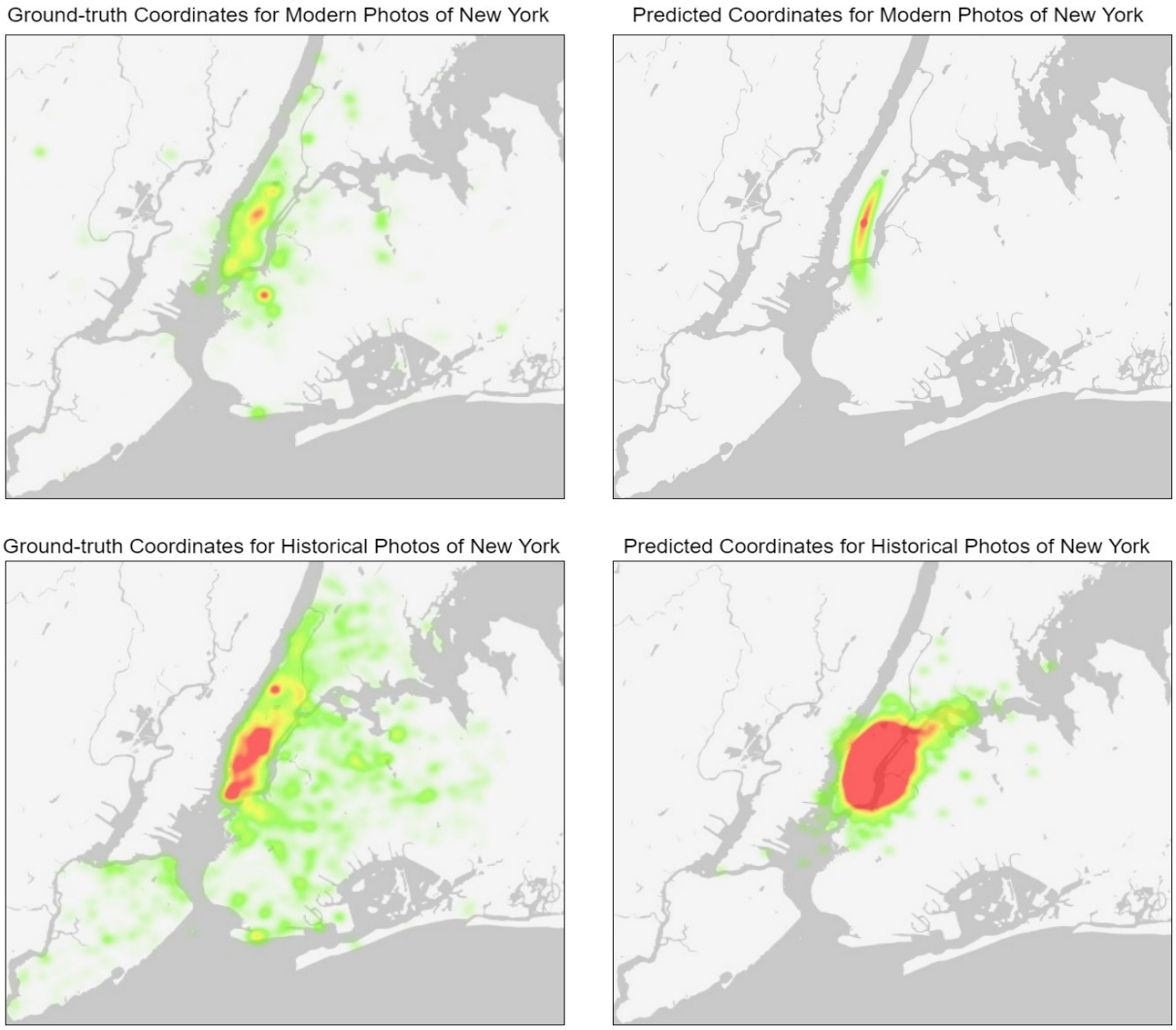


Figure 4.11: Density maps with basis on the ground-truth coordinates of the historical and modern photos from New York, versus the estimated coordinates by the neural networks using a regression loss.

wider area, with the trade-off of having higher distance errors.

Figure 4.10 illustrates the error distribution of the predicted coordinates in the best performing end-to-end network. Each dot represents an evaluation instance. Since the inferred coordinates by the end-to-end network not fine-tuned with historical data are concentrated in a single region, the heatmap with its error distribution is not conclusive. However, from the map in the left side of the figure, it can be concluded that the error increases as the photo's real coordinates get far from the area where the HEALPix most frequent cell is located. This is an expected result since all photos were assigned by the network to that area.

Table 4.5 reports the experimental results obtained with photos from New York. Starting with the results of evaluating modern and historical imagery directly on the geocoding networks also trained with modern and historical data, the regression networks of each set performed the

best in comparison with the other networks in the corresponding sets. Figure 4.11 illustrates the inferred coordinates distribution by the regression networks. Regarding the model trained with modern data, it is evident the concentration of the inferred coordinates in the same location where the most frequent HEALPix cell for this dataset is located. On the other hand, the model trained with historical data learned to predict photos in a much larger area, as the heatmap illustrates. Regarding the classification networks, resembling the experiments for the city of San Francisco, the predicted coordinates in both datasets were assigned to the cell containing the most photos, and the cells surrounding it. Regarding the combined networks, the one trained on the Flickr dataset predicted coordinates covering a much larger area, in comparison to the regression network, which may have also led to higher error distances, and thus making the results from the regression network the best ones. The covering area of the coordinates predicted by the combined network trained with historical data is similar to the distribution regarding regression network, which was expected as the metric's values of these two networks are similar.

To a better understanding of the image coloring component importance, similarly to the experiments with photos of San Francisco, there were first evaluated historical photos on networks trained in modern data from Flickr, and the results were compared with the ones obtained from the experiments in the introduced end-to-end network. Once again, it was proved that evaluating historical photos in networks trained with modern data produces better results when those old photos are first colored. Regarding the end-to-end network, experiments concluded that the network re-trained with historical data performs the best, contrary to what happens in the experiments on the end-to-end networks with San Francisco's datasets. The heatmap of the corresponding results distribution is illustrated in Figure 4.12. The results are more distributed, in comparison to the end-to-end network not fine-tuned with historical data, covering both overland and maritime regions, and are not uniquely concentrated around the HEALPix's most frequent cell. However, the error distribution from Figure 4.13 shows that the error is lower in the area of the HEALPix's cell containing the most photos, and increases significantly in the remaining areas of the city. Once again, the impact of having a big number of photos in a single region reflects on the achieved results.

None of the trained and evaluated end-to-end networks for both cities were capable of significantly outperforming the most frequent cell baseline. This is a solid baseline, as there is a significant number of instances in a single cell. When changing the resolution of the partitioning in order to get HEALPix cells with smaller dimensions, the baseline error values increased significantly in both datasets, as it is reported in both Tables 4.4 and 4.5. When using a coarse-

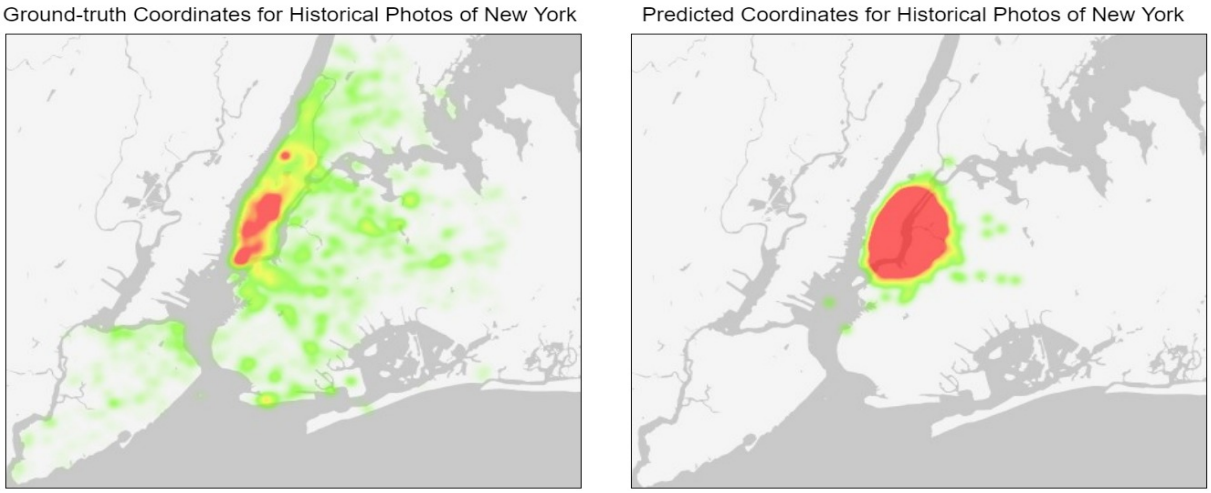


Figure 4.12: Density maps with basis on the ground-truth coordinates of the historical photos from New York, versus the estimated coordinates by a end-to-end network.

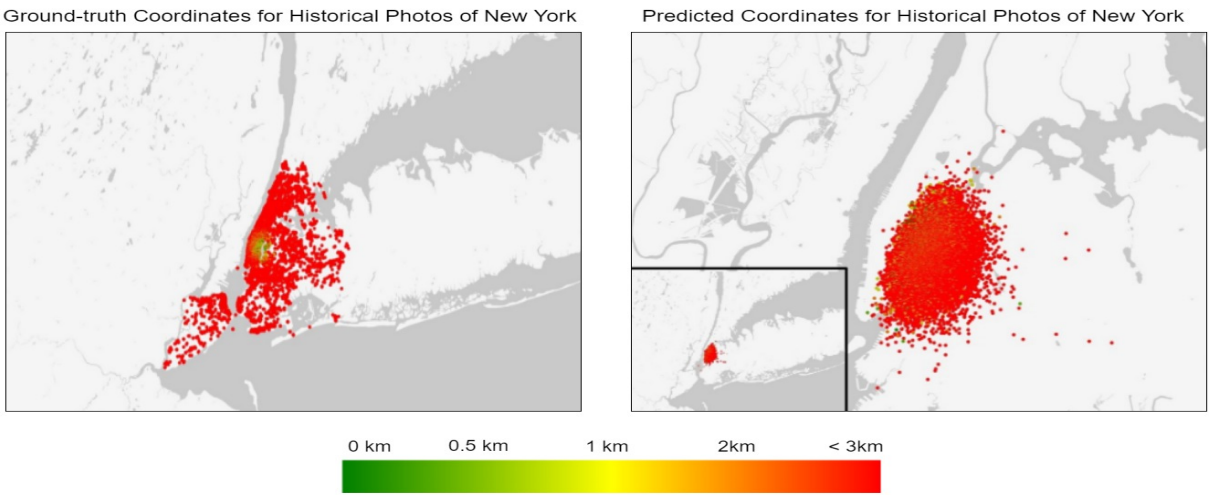


Figure 4.13: Error distribution with basis on the ground-truth coordinates of the historical photos from New York, and the estimated coordinates by a neural network combining regression and classification losses.

grained partition to get HEALPix cells with bigger dimensions, the error values also increase in comparison with the baselines using the same partitioning resolution as in the experiments, but not as much as when using the fine-grained partitioning. Table 4.6 presents additional information about these two new partitions, and establishes a comparison with the partition used in the experiments, referred to as *Current Partition*, in the same table. Using the coarse-grained partition would not solve the problem of having too many instances concentrated in a single cell, as the number of HEALPix cells is reduce significantly. However, using a fine-grained partition would indeed generate a more balanced distribution of the training instances over the cells, but at the same time Vo et al. (2017) in a case study concluded that the geolocalization accuracy gets worse if the partitioning is too fine.

Figure 4.14 and 4.15 present old photos concerning the cities of San Francisco and New

| | Current Partition | Coarse-grained Partition | Fine-grained Partition |
|---|---|---|---|
| Number of cells - Flickr San Francisco (SF) | 421 | 60 | 3415 |
| Number of photos in the most frequent cell - Flickr SF | 96595 | 511232 | 27733 |
| Number of cells - Historical (Hist.) SF | 63 | 9 | 491 |
| Number of photos in the most frequent cell - Hist. SF | 3156 | 8833 | 626 |
| Number of cells - Flickr New York (NY) | 820 | 70 | 2348 |
| Number of photos in the most frequent cell - Flickr NY | 135712 | 135 | 668 |
| Number of cells - Hist. NY | 328 | 37 | 2348 |
| Number of photos in the most frequent cell - Hist. NY | 3610 | 10772 | 668 |

Table 4.6: Analytical characterization of HEALPix partitions.

York, colored ($2^{nd}$ row) and geocoded with a distance error below 200 meters ($3^{rd}$ row) by the introduced end-to-end network. Each set of photos was geocoded by the end-to-end model that performed the best in the experiments in each city. The red and blue dots visible in the maps correspond to the ground-truth and the predicted locations of each photo, respectively, however given the low distance values, the two dots are not very distinguishable. These examples can help understanding which type of photos the proposed model can geocode the best. For the city of San Francisco, nine of the ten most well geocoded historical photos, i.e., the ones with the lowest distance error, correspond to outdoor pictures illustrating buildings. Concerning the photos of New York, nine of the ten most accurately geocoded photos correspond, once again, to outdoor areas, and all of those nine photos are also representing of buildings. Based on these pictures, one can say that both the fine-tuned and the not fine-tuned end-to-end networks were capable of learn to identify structures, such as buildings, and geocode them accurately.

## 4.4 Overview

This chapter specified the evaluation procedures used to conduct the experiments and to measure the results. There were also written two sections in this chapter reporting the results of those experiments: the first one detailed the experiments and the results of the coloring CNN, and the second one reported the experiments and the results of evaluating geocoding networks, with and without the coloring component attached to them. To define the baselines and conduct the experiments, there were leveraged the Yahoo Flickr Creative Commons 100 Million dataset, the Old San Francisco Project's dataset, and the Old New York Project's dataset.

Regarding the coloring model, there were trained and performed experiments on different networks with different configurations, such as using the MobileNet and the ResNet50 networks as feature extractors. The average RMSE quantitatively measured the efficiency of the trained models. The network that performed the best in the quantitative and qualitative evaluations was

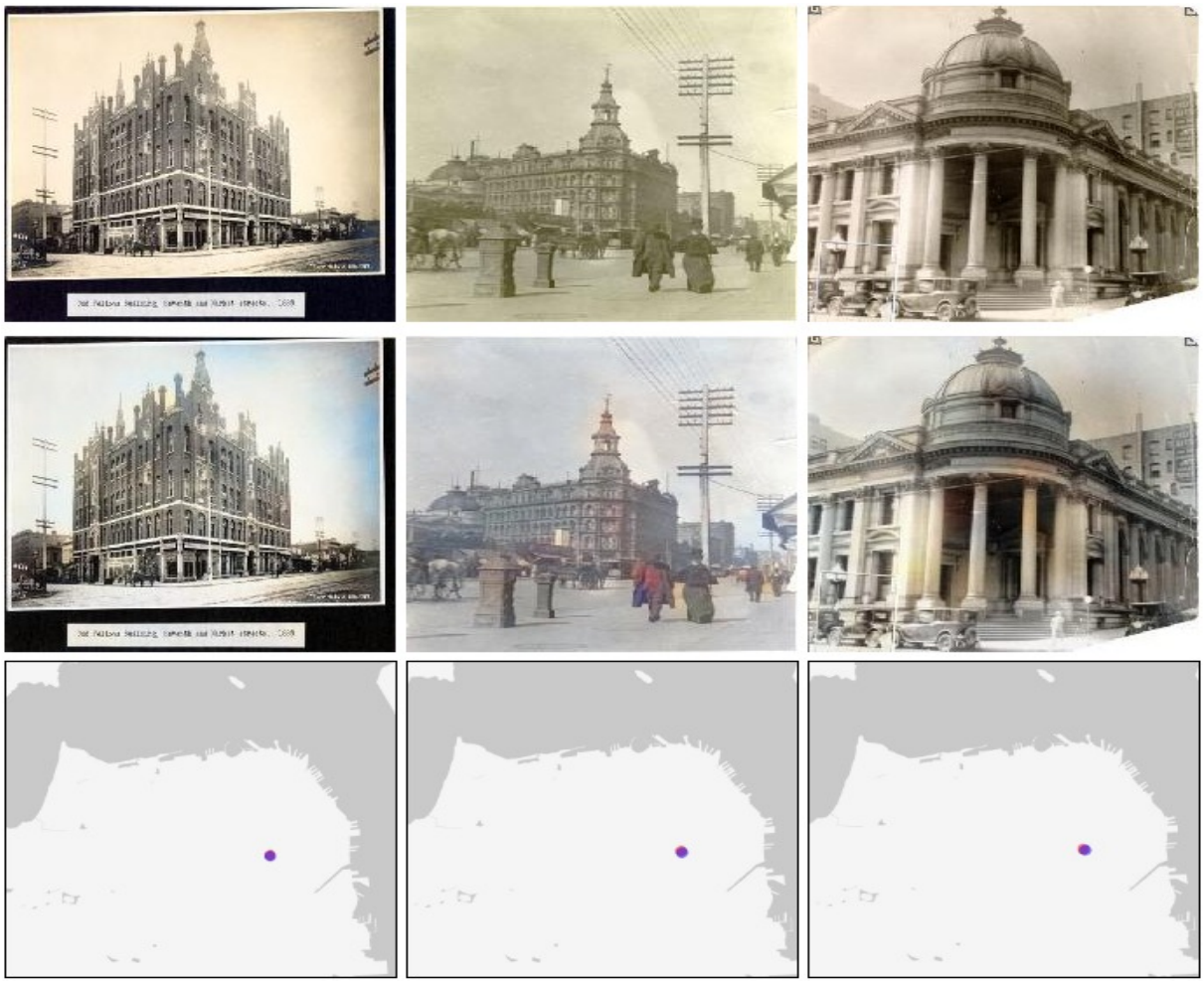

Figure 4.14: Best 3 geocoded historical photos from San Francisco.

the one using the ResNet50 as feature extractor, thus it was leveraged as the coloring component in the end-to-end geocoding network.

Concerning the end-to-end network, experiments showed that using the coloring component to geocode historical photos makes a major contribution to get better results in the task of geocoding historical photos using models pre-trained with modern data. Re-training the introduced end-to-end network with historical data improved the results in the dataset of old photos from New York, while for the dataset of old photos from San Francisco, not re-training the model achieved better results. The novel introduced end-to-end network was not able to outperform the *Hist. Most Frequent Cell* baseline, except in a single case where the difference between the median values was less than 100 meters. I argue that leveraging the ResNet50 in this end-to-end geocoding network was a good choice, since it is a deep network capable of producing good results in classification tasks, and in a case study the network significantly outperformed the InceptionResNetV2 architecture in both regression and classification tasks. The HEAPix partition resolution used in the experiments made some cells having a very high number of photos

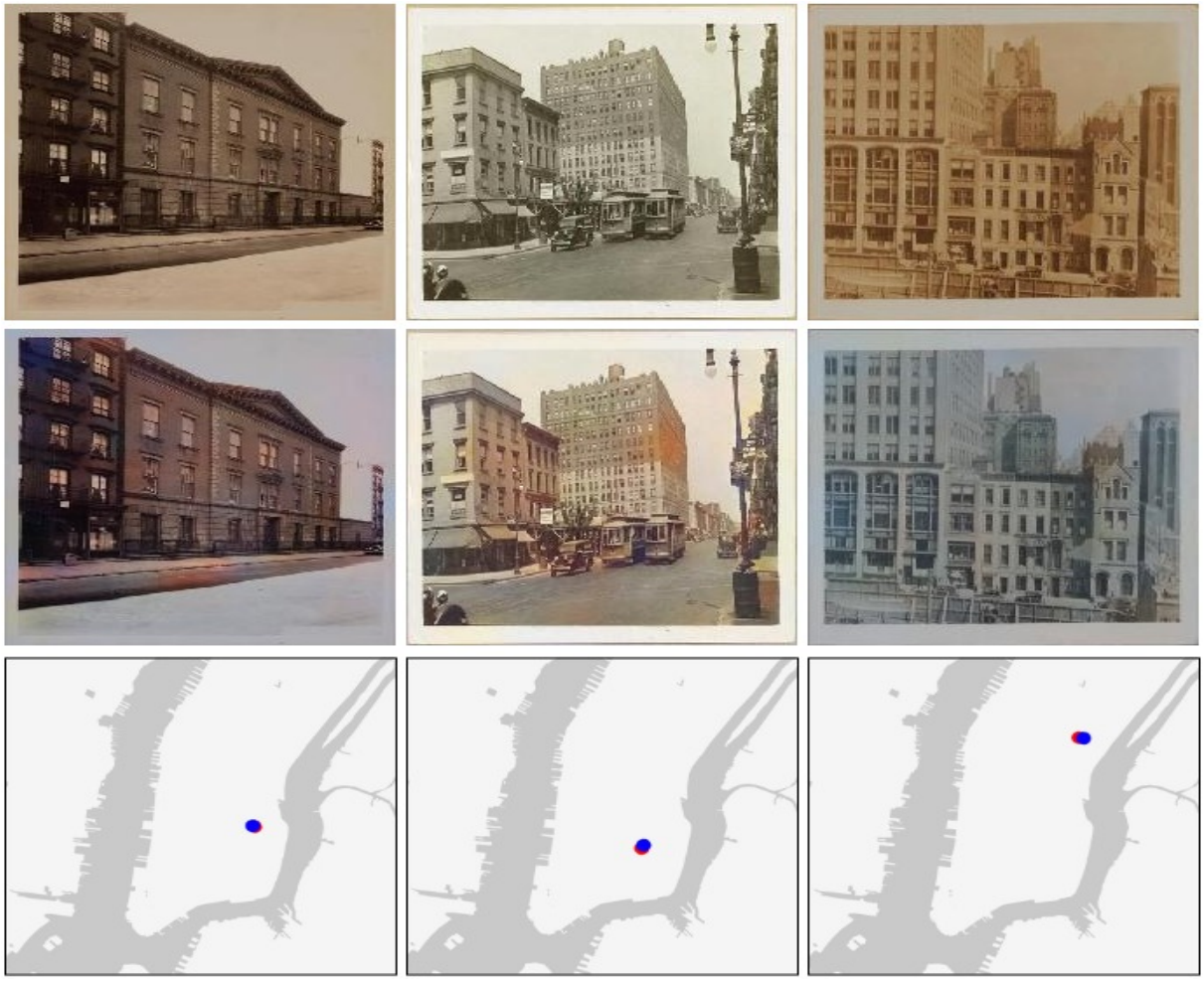

Figure 4.15: Best 3 geocoded historical photos from New York.

assign to them, causing an unbalanced distribution of the photos over those cells, which affected the results, as shown in this section. The introduced end-to-end network succeeds in geocoding outdoor historical photos of big structures such as buildings.

# 5 Conclusions and Future Work

This work presented an approach fully-based on CNNs, to address the task of geocoding historical photos automatically for the cities of New York and San Francisco. Considering the wide range of studies reported in the field of deep learning towards the tasks of geocoding photos and image transformations, it was introduced a novel end-to-end network. The network gathers a CNN trained to make color-wise image transformations, and a second CNN trained to geocode photos. In a single workflow, an old historical input image is colorized and then assigned to a geospatial location.

In this approach, the input given to the CNN trained to geocode photos is obtained from the CNN trained to colorized grayscale images. Thus, the success of the task is highly dependent on the performance of the colorizer component. However, just coloring old photos may not be enough to achieve an accurate geocoding, since the landscape changes over the years. For instance, an old photo of an open land may be, in the current landscape, a playground. In this scenario, coloring the old photo will not make a significant contribution in order to obtain an accurate result from the geocoding component. For this reason, and for many other reasons discussed during this dissertation, geocoding historical photos in the current landscape is an extremely challenging task, which reflected on the failure of the results, in comparison to the defined baselines. Nevertheless, results show that the introduced end-to-end network performs better in geocoding historical photos, in comparison to models trained with modern data that do not use the color component. Therefore, it is possible to argue that this end-to-end network is suitable to address the task, and with better resources it may outperform the baselines. For instance, expanding the size of the training set of the colorizer component, and filter the datasets used in the training of the geocoding networks in order to just use outdoor photos, may improve the results.

## 5.1 Future work

As future work, I believe it would be interesting to continue working on the introduced end-to-end network, fine-tuning the training resources as an attempt to improve the results.

As aforementioned, increasing the training dataset of the colorizer component, and filter the training datasets of the geocoding network concepts-wise, such as only use the outdoor and the sea pictures, could lead to an improvement of the results. Vo et al. (2017) in a case study trained a classification network using multiple Earth's partitions at the same time. If the same experiment is conducted in this work, it may improve the results of the networks using the classification loss function, as it, right now, they predict coordinates concentrated in unique small areas.

Regarding the colorizer component of the end-to-end network, experiments conducted showed that having a network capable of producing good colorized photos helps to improve the geocoding accuracy. Thus, in an attempt to obtain a better colorizer network, it would be interesting to replicate the approach used in He et al. (2018). In this work, given an input image and a reference image, the authors use first a CNN to infer the semantic similarity between the models, and a second network to color the pixels based on the similarity pairs of the first model. One other solution regarding the colorizer component, would be to train a network to perform this task as a classification one, similarly to Zhang et al. (2016), in which a grayscale input image is matched to a distribution of color values in the L*a*b* color space.

Concerning the architecture of networks trained to geocode photos, instead of making a drastic reduction in the feature vector, going from a dimensionality of 1000 directly to a dimensionality size of 2 (regression networks) or dimensionality equal to the number of HEALPix cells (classification networks), one possible alternative is to use Long Short-Term Memory (LSTM) units. Walch et al. (2017) use these units to perform dimensionality reduction, arguing that it leads to an increase in the localization performance.

# Bibliography


Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Chen, T. Q. and Schmidt, M. (2016). Fast Patch-based Style Transfer of Arbitrary Style. *arXiv preprint arXiv:1612.04337*, (1).

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. (2017). StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. *arXiv preprint arXiv:1711.09020*.

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Deshpande, A., Rock, J., and Forsyth, D. (2015). Learning large-scale automatic image colorization. In *Proceedings of the IEEE International Conference on Computer Vision*.

Federico Baldassarre, Diego Gonzalez-Morin, L. R.-G. (2017). Deep-koalarization: Image colorization using cnns and inception-resnet-v2. *ArXiv:1712.03400*, (1).

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image Style Transfer using Convolutional Neural Networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Gorski, K. M., Felten, J. E., Wandelt, B. D., Hansen, F. K., Hivon, E., and Banday, A. J. (1999). The HEALPix Primer. *arXiv.org*, astro-ph(2).

Hays, J. and Efros, A. A. (2008). IM2GPS: Estimating Geographic Information from a Single Image. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Hays, J. and Efros, A. A. (2015). Large-Scale Image Geolocalization. In *Multimodal Location Estimation of Videos and Images*. Springer.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceeding of the IEEE International Conference on Computer Vision and Pattern Recognition*.

He, M., Chen, D., Liao, J., Sander, P. V., and Yuan, L. (2018). Deep Exemplar-based Colorization. *ACM Transactions on Graphics*, 37(4).

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. In *Proceedings of the NIPS Deep Learning and Representation Learning*.

Hochreiter, Sepp and Schmidhuber, Jürgen (1997). Long Short-Term Memory. *Neural Computation*, 9(8).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, abs/1704.04861(1).

Iizuka, S., Simo-Serra, E., and Ishikawa, H. (2016). Let There Be Color!: Joint End-to-End Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics*, 35(4).

Isola, P., Zhu, J., Zhou, T., and Efros, A. A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. *CoRR*, abs/1611.07004(1).

Jegou, H., Douze, M., and Schmid, C. (2008). Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Proceedings of the European Conference on Computer Vision*.

Johns, J. M., Rounds, J., and Henry, M. J. (2017). Multi-modal Geolocation Estimation Using Deep Neural Networks. *CoRR*, abs/1712.09458(1).

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proceedings of the European Conference on Computer Vision*.

Krizhevsky, A. and Hinton, G. (2009). Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the Conference on Neural Information Processing Systems.*

Larsson, G., Maire, M., and Shakhnarovich, G. (2016). Learning Representations for Automatic Colorization. In *Proceedings of the European Conference on Computer Vision.*

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based Learning Applied to Document Recognition. In *Proceedings of the IEEE.*

Levin, A., Lischinski, D., and Weiss, Y. (2008). A Closed Form Solution to Natural Image Matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2).

Li, C. and Wand, M. (2016). Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition.*

Liu, M.-Y. and Tuzel, O. (2016). Coupled Generative Adversarial Networks. In *Advances in Neural Information Processing Systems.*

Luan, F., Paris, S., Shechtman, E., and Bala, K. (2017). Deep photo style transfer. *CoRR, abs/1703.07511*, 2(1).

Radenovic, F., Tolias, G., and Chum, O. (2016). CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition.*

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for Activation Functions. *CoRR*, abs/1710.05941(1).

Rawat, W. and Wang, Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural computation*, 29(9).

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*, 115(3).

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN Features off-the-shelf: an Astounding Baseline for Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.*

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. (2017). Learning from Simulated and Unsupervised Images through Adversarial Training. In *CVPR*.

Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, (1).

Song, H. O., Xiang, Y., Jegelka, S., and Savarese, S. (2016). Deep Metric Learning via Lifted Structured Feature Embedding. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going Deeper with Convolutions. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In *Proceeding of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. (2016). YFCC100M: The New Data in Multimedia Research. *Communications of the ACM*, 59(2).

Tian, Y., Chen, C., and Shah, M. (2017). Cross-View Image Matching for Geo-localization in Urban Environments. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

Tsioukas, V., Koussoulakou, A., and Jancsó, T. (2015). Falling into Place: Orientation and Viewing of Past Cityscapes in 3D Space. *e-Perimetron*, 10(1).

Vincenty, T. (1975). Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations. *Survey Review*, 23(1).

Vo, N., Jacobs, N., and Hays, J. (2017). Revisiting IM2GPS in the Deep Learning Era. In *Proceedings of the IEEE International Conference on Computer Vision*.

Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., and Cremers, D. (2017). Image-based Localization using LSTMs for Structured Feature Correlation. In *Proceedings of the IEEE International Conference on Computer Vision*.

Weyand, T., Kostrikov, I., and Philbin, J. (2016). Planet-Photo Geolocation with Convolutional Neural Networks. In *European Conference on Computer Vision*.

Workman, S. and Jacobs, N. (2015). On the Location Dependence of Convolutional Neural Network Features. In *Proceedings of the Workshops at the IEEE International Conference on Computer Vision and Pattern Recognition*.

Workman, S., Souvenir, R., and Jacobs, N. (2015). Wide-Area Image Geolocalization with Aerial Reference Imagery. In *Proceedings of the IEEE International Conference on Computer Vision*.

Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful Image Colorization. In *Proceedings of the European Conference on Computer Vision*.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning Deep Features for Scene Recognition using Places Database. In *Proceedings of the Conference on Neural Information Processing Systems*.

Zhu, J., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv preprint arXiv:1703.10593*, (1).

Zhu, X. and Bain, M. (2017). B-CNN:Branch Convolutional Neural Network for Hierarchical Classification. *CoRR*, abs/1709.09890(1).