Automatic User Profiling for Intelligent Tourist Trip Personalisation

Liam Attard

Supervisor(s): Dr Josef Bajada

Faculty of ICT

University of Malta

June 2021

Submitted in partial fulfillment of the requirements for the degree of B.Sc. ICT in Artificial Intelligence (Hons.)

L-Università ta' Malta

FACULTY/INSTITUTE/CENTRE/SCHOOL_____

DECLARATIONS BY UNDERGRADUATE STUDENTS

Student's I.D. /Code _____

Student's Name & Surname _____

Course _____

Title of Long Essay/Dissertation

_____

_____

_____

Word Count _____

(a) Authenticity of Long Essay/Dissertation

I hereby declare that I am the legitimate author of this Long Essay/Dissertation and that it is my original work.

No portion of this work has been submitted in support of an application for another degree or qualification of this or any other university or institution of higher education.

I hold the University of Malta harmless against any third party claims with regard to copyright violation, breach of confidentiality, defamation and any other third party right infringement.

(b) Research Code of Practice and Ethics Review Procedures

I declare that I have abided by the University's Research Ethics Review Procedures.

_____ _____

Signature of Student Name of Student (in Caps)

_____

Date

08.02.2018

Abstract:

Sample abstract.

Acknowledgements:

Sample Acknowledgement

Contents

List of Figures

List of Tables

1 Introduction

Planning for a trip is a challenging task due to the variety of real-life constraints. This chapter will discuss the main research problem confronting automatic tourist planning along with this dissertation's objectives and proposed solution.

1.1 Problem Definition

Leisure travelling is an impactful industry whose economic importance significantly improves each year, contributing to 10.4% of the global GDP in 2019 [1]. Despite this, planning for a trip to a foreign city requires a substantial amount of time-consuming research. As a result, people often rely on multiple data sources such as travel brochures, blogs and vlogs to form a holiday plan and retrieve the top-rated points of interests (POI) of a site. However, a tourist has to compile a timetable independently even though mediums do not hold the resources to provide POIs tailored according to the traveller's preferences and constraints [2].

In literature, offering tourists a personalised route composed of POIs has been defined as the tourist trip design problem (TTDP). The TTDP comprises ranking and selecting POIs that might interest the user and create a feasible plan. Figure 1 shows an example of the TTDP, where a tourist has to form a timetable that balances between the POI's rating and location while satisfying the various trip constraints. The TTDP is an NP-hard problem where complete algorithms only manage to optimise with a small number of POIs. Therefore, many approximate algorithms, namely heuristic and meta-heuristic approaches, work to converge solutions with complex alternatives to this problem.Section 2 provides a detailed review of this problem and its variants.

Nevertheless, the few existing systems that provide users with an itinerary or route require a lengthy process of manually gathering the users' likes and constraints or information from past trips. Therefore, we will try to answer the following research question:

Can a system automatically get to know what a tourist likes to visit and use this information to generate a personalised itinerary for a holiday?

1.2 Motivation

Our primary motivation behind this work is to introduce the automatic retrieval of user preferences for other travel planning applications. Currently, there is no mainstream ap-

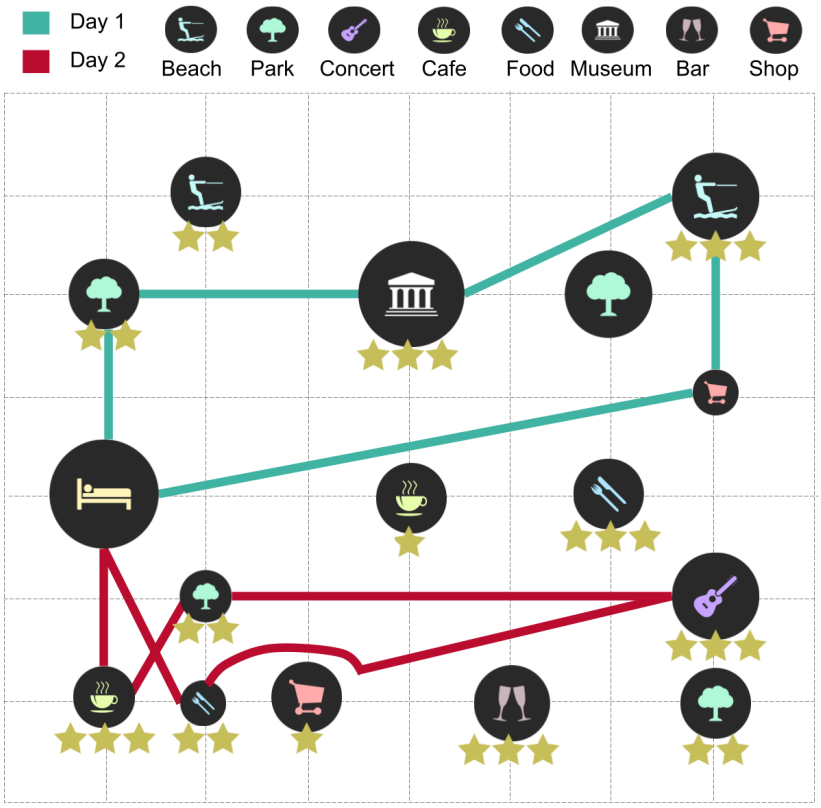

Figure 1: Example of a tourist planning problem

plication that provides tourists with a fully personalised plan from a quick and easy to use application. We were also motivated by the idea of delivering one centralised system which organises the whole holiday rather than having to spend time searching through the excessive amount of data online. This approach is better than bombarding potential tourists with many questions to understand the users' personalities.

Existing algorithms and tourist planners use heuristics to optimise solutions for the timetable problem and achievable results in polynomial time [3].

Collecting the users' preferences is a beneficial technique used by businesses to advertise their products by targeting only a specific audience [4].

In this dissertation we combine both technologies to provide one system.

1.3 Aims and Objectives

This dissertation aims to build an application that generates a personalised holiday plan according to the user's travel dates and constraints.

- **Objective 1 (O1)**: Investigate techniques to build travel interest profiles automatically from social media interactions.
- **Objective 2 (O2)**: Explore different optimisation algorithms for building personalised travel itineraries using the generated travel interest profiles.
- **Objective 3 (O3)**: Evaluate the performance of the personalised travel itinerary generator with real users through in-depth semi-structured interviews.

We will conduct the interviews by generating a personalised and non-personalised timetable for a holiday in Malta to have prior knowledge of the POIs and compare the effects of both itineraries.

1.4 Proposed Solution

To address this problem, we present an application that helps tourists travel by providing them with a complete itinerary for their upcoming holiday using several optimisation algorithms, namely, Genetic Algorithms (GA) and Particle Swarm Optimisation (PSO).

With the prevalence of social media and data-driven approaches, we also automate gathering users' POI desires by scanning their social media profile using machine learning classification approaches through Convolutional Neural Networks (CNN).

The results from the evaluation section 4 show that we were able to classify the user's photos and liked pages into a travel interest vector that automatically describes the user's characteristics. We were also able to provide optimisation techniques that converge to the timetables with the best scores given a set of tourist constraints. In-depth semi-structured interviews with several participants continued to provide us with further detail on the accuracy of the user profiling algorithm and the resulting timetables.

1.5 Document Structure

This dissertation is structured as follows; Section 2 discusses related work relating to existing TTDP solutions and automatic user preference gathering. Section 3 demonstrates the steps taken to create the whole application along with its underlying mechanism. Section 4 will evaluate the performance of the convolutional neural networks, the optimisation algorithms and discuss the interview's outcomes. Finally, section 5 will address the findings obtained from this research concerning the objectives and some future improvements.

2 Background and Literature Review

This section aims to position our study by discussing existing automatic user preference gathering techniques and tourist planning solutions. In the first part, we discuss user profiling to represent travel preferences. We then give an overview and formalise the Tourist Trip Design Problem (TTDP) research area. Finally, we discuss the gaps in the current TTDP literature that this dissertation will address.

2.1 Automatic User Preference Gathering

User profiles are a virtual representation of a user containing their characteristics [5]. In addition, some tourist planners make use of such a technique to personalise the results of their system [6–9]. For example, Wörndl et al. [6] required the upcoming tourists to input their preferences manually by rating six categories on a scale of 0 to 5: *Sights and Museums, Night Life, Food, Outdoors and Recreation and Shopping.* Including a manual input of user preferences resulted in high user satisfaction since their timetable was very customised.

2.1.1 Automatic preference gathering in Travel Planning Applications

In 2018, Lim et al. [7] demonstrated how implementing personalisation in their algorithm, PersTours, helped portray real-life scenarios more accurately. The authors built a system where the tourist's level of interest in a specific category is dependant on their time spent at such POIs, relative to the average user. First, they gathered information from the user's past trips from the social media platform Flickr. Then, they evaluated their algorithm using the Root-Mean-Square Error (RMSE), representing the time deviation of past trips and PersTours results from Flickr. Although their results show the PersTours outperforms other applications that use frequency-based user interest, this approach requires users to use Flickr and post information about their past trips on the platform.

Nguyen et al. [10] developed an Android chat application called STSGroup that gathers user's preferences and resolves conflicts between tourists by understanding the messages sent in a group chat. They provided an example of students travelling to South Tyrol (Italy), which gathered information such as the users' mood and recommended POIs from their conversations. Other users in the group chat rate their suggestions through a voting system as the system uses raking and logistics to calculate the ideal group preferences in the background. As a result, 86.7% of the test users showed satisfaction with the suggestions.

2.1.2 Automatic preference gathering through social media

The average internet user has gone from being a passive content absorber to a content producer through social media. TTDP solutions can use this advantage and provide a fully automated activity plan based on the user's characteristics. The following are some methods for user profiling and information gathering from the user's social media.

Instagram has a significant effect on the tourism industry. Sharing photos of amazing sights and landscapes influence the way people choose their POIs [11]. Therefore, a system that uses tourist's social media photos could infer the user's preference.

Guntuku et al. [12] performed an analysis on the relationship between a user's characteristics and online images. They found that the media on the social media profile can predict the big five personality traits; conscientiousness, extraversion, neuroticism, agreeableness and openness. The performance graded by the Pearson correlations tests were 0.530 and 0.566 for prognosticating neuroticism and conscientiousness, respectively.

Chen et al. [13] produced a system for automatically retrieving tags from images and incomplete tags called *FastTag*. The algorithm uses two simple linear mappings. Figure 2 shows an example of an input image used alongside the tags *snow, lake, and feet*. In addition, the algorithm produced the tags; *mountain, water, legs, boat, trees*.

Figure 2: Example of an image input for the FastTag algorithm

These approaches show how image classification techniques could provide an automatic preference gathering system.

2.2 Travel Planners for both individual and grouped travellers

This section will analyse existing optimisation techniques for TTDP solutions. These include swarm-based, trajectory-based and evolutionary algorithms. Gavalas et al. [14]

classify TTDP variants into two; Systems that produce a single route and systems that can handle multiple days.

Trip planner applications aim to offer tourists information in a unified and centralised manner, providing them with a plan for their trip. Two domains develop current applications: methods for obtaining POIs and tour recommendation algorithms that create tourist trips, as shown in Figure 3.

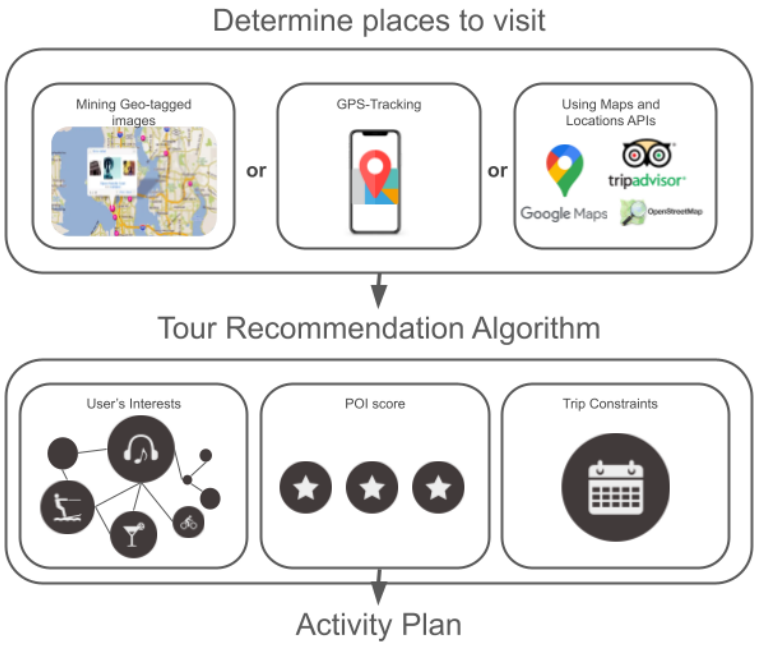

Figure 3: Trip Planner Applications Process.

2.2.1 Collecting the Points of Interests

Before producing an itinerary, the tourist planners have to formulate a dataset of POIs from some data source. There are several ways to identify an appropriate data source representing real-life tourist trajectories.

One approach is made by gathering tourist places by mining them from geotagged images of Location-Based Social Networks (LSBN) such as Flickr, or Facebook [2,7,15–22].

The ubiquitous presence of smartphones and GPS-enabled devices could help gather the best POIs to visit based on other users' historical paths [23–25]. However, privacy issues are the main caveat towards this approach since it requires people to share their location constantly and publically [26].

A prompt and accurate strategy towards gathering essential places in the vicinity uses Mapping & Location APIs such as Foursquare, Google or TripAdvisor. Worndl et al. [6] use this approach and build a dataset of prominent POIs by querying their API with the user's desired location. In return, they receive a sequence of places and information about each site, including its category, other user's ratings, opening hours, coordinates and helpful additional information to use as criteria for the itineraries.

2.2.2 Single Route Problems

The Orienteering Problem (OP), introduced by Tsiligirdes [27], is the foundation of single route planners in observance of the sport, orienteering. There are various types of OPs that include different constraints, such as time windows and time dependency, as shown in figure 4. The OP can be represented as a travelling salesman problem with profits. Gavalas et al. and Vansteenwegen et al. [14, 28] mathematically formulate the OP as follows:

Consider a set of POIs

$$P = p_1, p_2, \ldots, p_N$$

where:

p:	A POI having the latitude, longitude, category, price, and corresponding set of opening hour constraints

The objective function of OP is:

$$\text{MAX} \sum_{i=2}^{N-1} \sum_{j=2}^{N} s_i x_{ij}$$

where:

s_i	score of visiting node i
x_{ij}	if a visit by POI i is followed by POI j (0 otherwise)

Constraints

$$\sum_{j=2}^{N} p_1 j = \sum_{i=1}^{N-1} p_{iN} = 1 \qquad (1) \qquad\qquad \sum_{i=1}^{N-1} \sum_{j=2}^{N} t_i x_{ij} <= Tmax \qquad (2)$$

$$\sum_{i=1}^{N-1} x_{ir} = \sum_{j=2}^{N} x_{ij} \geq 1 \,\mathrm{r} = 2, \ldots, \mathrm{N\text{-}1} \tag{3}$$

$$2 \geq u_i \geq N \tag{4}$$

$$u_i - u_j + 1 \geq (N-1)(1 - x_{ij}) \tag{5}$$

where:

t_{ij}	travel time from i to j
T	maximum time
u_i	position of POI i in route

Constraint 1 ensures that the path starts at POI 1 and ends at POI N. Constraint 2 limits the total travel time. Constraint 3 ensures that each POI is visited only once. Constraint 4 and constraint 5 ensures that there are no subtours.

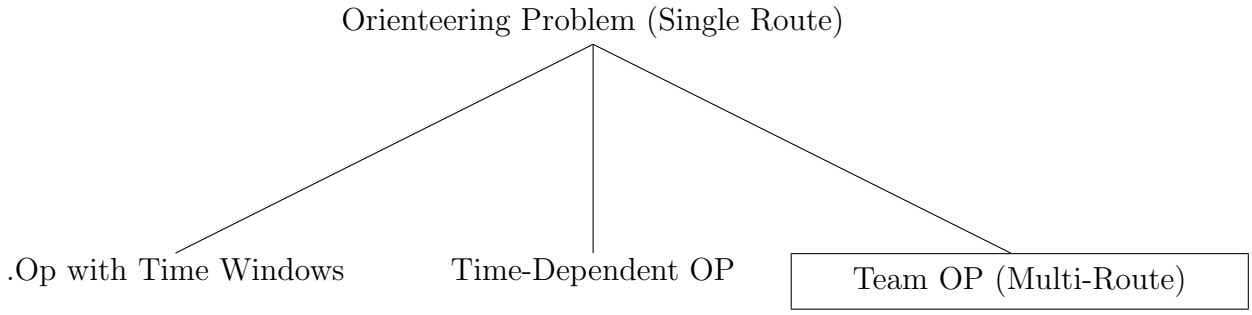

Figure 4: Some Variants of the Orienteering Problem

There are numerous Evolutionary Algorithms (EA) proposed to solve OP. [29, 30]. EAs are algorithms based on natural evolution which use a fitness score to get to the best solution of a problem, in this case, the TTDP [31].

Particle Swarm Optimisation-based (PSO) systems provide prevalent OP solutions with fast computing time [32]. These are bio-inspired meta-heuristic approaches in which, in the TTDP, a particle represents a travel path. The particles aim to optimise themselves by communicating with each other and using their velocity property to move to the most optimal solution [33]. Sevkli et al. [34, 35] tested out two PSO variants: Strengthened Particle Swarm Optimization (StPSO) and Discrete Strengthened Particle Swarm Optimization

(DStPSO). These two algorithms introduce pioneering particles, which first perform a local search-based technique called Reduce Variable Neighborhood Search (RVNS) between all the particles and then assign a random velocity. These PSO algorithms obtains either the best or competitive solutions compared with other algorithms such as Ant Colony and Genetic Algorithms when tested on the Tsiligirides [25, 27] dataset of predefined nodes.

A novel approach in 2018 by Kobeaga et al. [29] was able to achieve competitive solutions for medium-sized instances of over 400 nodes and find new best-known solutions for large datasets using the steady-state genetic algorithm. The algorithm also implements a local search, which aims to reduce travel time.

In 2019, Santini et al. [36] introduced a heuristic algorithm based on adaptive extensive neighbourhood search. They evaluated their system by comparing it with Kobeage et al.'s EA. The results showed that both algorithms find competing solutions in a reasonable amount of time. However, the EA finds slightly more suitable solutions, while the extensive neighbourhood search has a lower average gap.

In real-life scenarios, POIs have time constraints that allow them to be visited only during specific hours, such as opening and closing hours or public holiday constraints. Traditional OP is not able to cater for such problems. A single route variant of the OP which solves these issues is the Orienteering Problem with Time Windows (OPTW) [14].

Kantor et al. [37] provided the first attempt towards the OPTW [3]. They developed two algorithms; Insertion and depth-first search. The former algorithm solves the path by selecting a POI with the highest score over-insertion cost incrementally. On the other hand, the depth-first search algorithm gathers parallel tree-based solutions simultaneously and iteratively adds new POIs as long as they follow a set of constraints. Their evaluation showed significant improvements of the second algorithm over the insertion.

When travelling between two POIs, the travel time may depend on certain variable time constraints such as the traffic levels and waiting time [38]. The Time-Dependent Orienteering Problem (TDOP) introduced by Fomin et al. [39] is the single route variant of OP, which considers these scenarios since traditional OPand OPTW does not [31]. In 2011, Abbaspour et al. [40]provide a solution for the Time-Dependent Orienteering Problem with Time Windows, which combines the two previously mentioned OP variants (TDOPTW). They propose two adaptive genetic algorithms and multi-modal shortest pathfinding evaluated in the city of Tehran.

2.2.3 Multiple Route Problems.

The solutions available from what we discussed in the previous sections can only generate a single efficient path for a tourist's holiday. The Team Orienteering Problem (TOP) [41] is a variant of the OP, which allows for solving the TTDP with multiple days [42]. The system generates a full itinerary for the tourist, with a maximum total score of all routes [38].

Several solutions use PSO-based algorithms to solve the TOP [32,43,44] Muthuswamy et al. [43] developed a discrete version of the PSO (DPSO) which can generate n routes where $2 \geq n \geq 4$. The algorithm consists of two procedures; Random initialisation of n-1 routes. The n^{th} route is based on partial randomness and the current score divided by the current distance of each particle after updating the velocity. The particles use RVNS and 2-opt techniques to communicate with each other as local search techniques. The authors evaluated their work by comparing the algorithm to seven TOP heuristics in which DPSO performed competitively across all applied benchmark data sets [14].

A few years later, Dang et al. wrote another PSO inspired algorithm (PSOiA) for the TOP. They evaluated their work using an interval graph model, which showed how to examine a more extensive search space faster [31].

Besides swarm-based algorithms, an algorithm by Sylejmani et al. [45] used the trajectory-based tabu search to solve a Multi Constrained Team OPTW. Their system followed three steps in order to generate an activity plan: a new activity is added as a node to the trip using *Insert*, a node is exchanged with a new activity using *Replace* and two nodes swap with each other using *Swap*.

2.3 Conclusion

We have concluded that it is possible to gather characteristics from social media throughout this research. Therefore, we will introduce this technology to generate user profiles as part of a constraint with the objective function of the optimisation algorithm.

Since the evolutionary algorithms, PSO and GA, resulted in competing solutions on the Tsilgidres Dataset and are even on novel solutions [32,44], we will compare both algorithms to see which one to use as the baseline for our TDOPTW application.

3 Methodology

This section will elaborate user-profiling methods, the itinerary generator and the implementation used to build this application. Figure 5 outlines the overall process of our personalised itinerary generation framework.

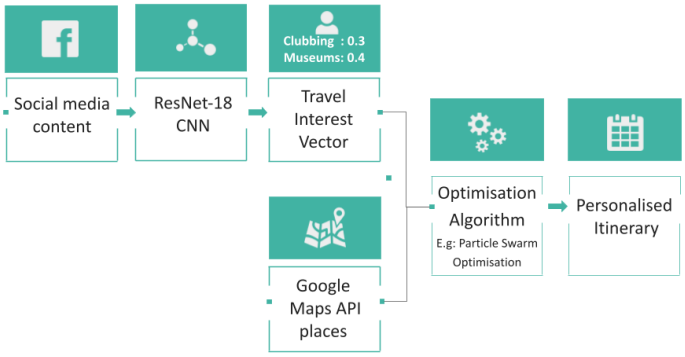

Figure 5: Personalised itinerary generator

3.1 Generating the User Profile

Many people are present on social media posting pictures of their travel moments [46]. For this reason, we opted to build a user profile from the user's online presence.

A travel interest vector is made up a vector of six integer values

$$< v_0, v_1, v_2, v_3, v_4, v_5 >$$

representing the user profile. Each element represents the following POI categories respectively:

$$< 0 \text{ Beach}, 1 \text{ Nature}, 2 \text{ Shopping}, 3 \text{ Museums}, 4 \text{ Clubbing}, 5 \text{ Bars} >$$

At the start of the application, the travel interest vector is initialised with zero values, and the app increments a category whenever the user's content matches. v_0 to v_3 represents morning categories and v_4, v_5 represent evening categories. After the data collection process is complete, the morning and evening vector values are normalised independently.

Facebook's API that allows users to connect both their Facebook and Instagram accounts and request content from the user with their permission. The app requests the photos and the liked pages used to populate the user's travel interest vector.

3.1.1 Transforming the liked pages into the travel interest vector

The API's documentation contains a whole list of possible page categories. The app iterates through all of these user's liked page categories and increments a value in the travel interest vector whenever the Facebook result matches. For example, if a user likes a page with class 'DJ', the user's clubbing vector value is incremented, and if a page is labelled as a 'Mountain', the app increments the user's nature vector value.

3.1.2 Transforming the user's photos into the travel interest vector

Convolutional Neural Networks have become a standard for classifying an image because of their high accuracy [47]. Therefore, we decided to test out two approaches for classifying the photos into the app's six categories.

Zhou et al. [47] trained several CNNs for scene recognition and generic deep scene features for visual identification. However, the places365 models are not explicitly trained on the six categories of our application. Therefore, we need to carefully map the 365 categories with our six application's categories. That is why we introduced a Tensorflow Keras sequential model, explicitly trained on the six application's categories to compare.

Pretrained Places365 models: These models are trained on the places365-standard dataset of about 1.8 million images to classify an image into 365 different scene categories. We used the Resnet places365 models, Resnet-18 and Resnet-50 since they achieved the highest top-5 validation accuracy on the places365 dataset. The Resnet 18 comprises 18, and the Resnet 50 comprises 50 convolutional layers. They both converge to an output layer representing the 365 output categories. Figure 7 shows a summary of the whole Resnet 18 model architecture.

Trained Tensorflow Keras model: This model contains three convolutional layers with a rectified linear unit (ReLu) activation function. A pooling layer follows each to lower the input volume's spatial dimension for the upcoming layers. The first layer is a rescaling layer that resizes an image to 180×180 pixels. The final layer represents a flattening layer and two dense layers to reduce the outputs to the six application categories, and another representing the 'None' classification. Figure 7 shows a summary of the whole model.

The dataset contains 3600 public internet images representing the seven classes: Beach, Nature, Museums, Shopping, Clubbing and Bars and None. The tensor library provides tools to Split the dataset into a training and validation set Distribute the photos into

Figure 6: Data augmentation to reduce overfitting

batches of 32 Cache the dataset to memory to prevent I/O blocking All of the images were resized to 180x180 pixels, and the RGB values were normalised from zero to one. Since the dataset is small compared to the places 365 models, the training process is prone to overfitting. Data augmentation generates additional samples using random transformations on the dataset. Figure 6 shows an example of data augmentation on a photo. We also added a dropout layer to the model randomly drops sets the input values of the neuron. These two techniques help the model avoid overfitting.

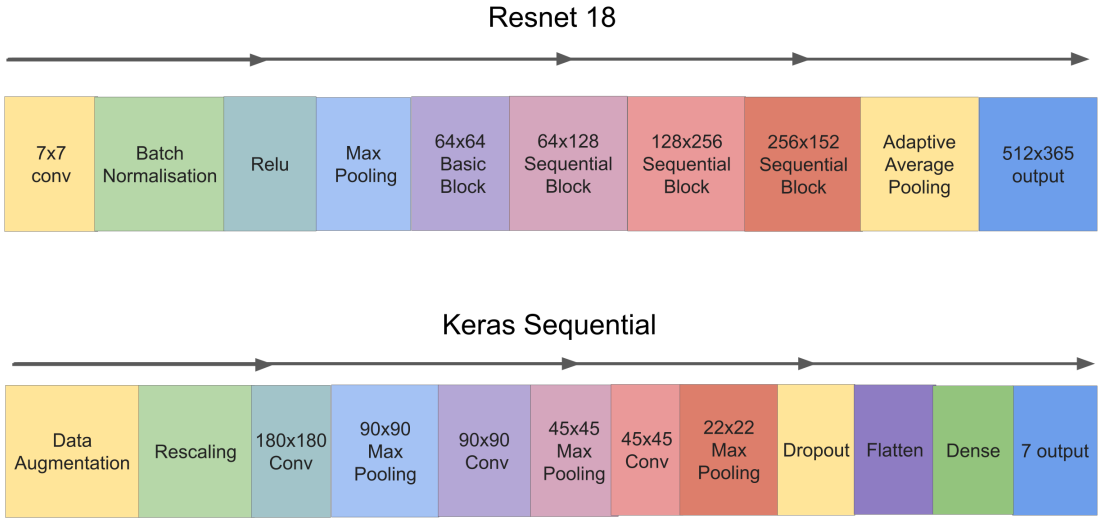

Figure 7: Model Architectures

3.2 Producing the activity plan

After the app generates the dataset of POIs and the user's travel interest vector, we formulate an efficient activity plan using these two inputs. This itinerary generator is based on the existing state of the art activity planners [42, 44] with some adjustments: We wanted the trip's output to take the form of an itinerary. The problem takes the form of the 'day' and 'night' category split discussed in the literature review. The scoring of itineraries is adjusted with the travel interest vector.

The problem definition of our novel itinerary planner is mathematically formulated as follows. A tourist trip is made up of some pre-defined user constants alongside the travel interest vector. The predefined constants are:

M:	The number of travelling days
C:	The activity pace (ie. the greater the C value, the more activities are generated in a day)

The objective function of our itinerary planner is:

$$\text{MAX} \sum_{m=0}^{M} (S_{D_m} + S_{E_m})$$

where:

m	Travelling day (m=1,2,..., textitM)
D_m	Morning section of day number m
E_m	Evening section of day number m
S_{D_m}	Score of the morning section D_m
S_{E_m}	Score of the evening section E_m

A day is made up of the morning D_m section and the evening E_m section. The timetable suggests a POI in the morning, then somewhere to eat, and the rest is dependant on the activity pace C. That is why the morning section is made up of $C + 2$ tourist attractions. The evening section suggests a place to eat and a POI; therefore, the evening section is just made up of 2.

$$D_m = Y_i + Y_f + C(Y_i)$$

$$E_m = Y_f + Y_j$$

i	Morning Tourist attraction (i = 1, 2, 3,. . . , n_1)
j	Evening Tourist attraction (j = 1,2, 3,. . . , n_2)
f	Food Place (f = 1,2, 3,. . . , n_3)
$Y_{i\|f\|j}$:	1 if a tourist visit attraction i, j or f and 0 if otherwise

Constraints

$\sum_{m=0}^{M} \sum_{i=0}^{n_1} Y_i \leq 1$	Ensures that all morning tourist attractions are not visited more than once throughout the whole itinerary
$\sum_{m=0}^{M} \sum_{j=0}^{n_1} Y_j \leq 1$	Ensures that all evening tourist attractions are not visited only once throughout the whole itinerary

3.2.1 Calculation of Score

The score S_{D_m} or S_{E_m} is calculated using

$$S_{D_m|E_m} = \frac{1}{T} + R + V$$

where:

T	Total distance between each tourist attractions in the morning/evening of day m
R	Average rating of the tourist attractions in the morning/evening of day m
V	how much the tourist attractions of the morning/evening of day m match with the user's travel interest vector

3.2.2 Retrieving the Points of Interests Dataset

We used the Google Maps API as the data source for our application because of its accuracy and updated dataset compared with the other approaches and other APIs [48, 49]. In addition, the nearby search endpoint allows the app to search for places of a given category within a specified area. In order to retrieve the places for the application, eight requests are made, each requesting places of different categories. To solve the issues with time

Table 1: Sample Query being made to the google maps nearby search endpoint

Key	Value
location	35.93575, 14.3754
radius	50000
type	cafe
keyword	must visit tourist

windows, we split the endpoints into two categories. Five of the requests represent places shown as part of the itinerary during the day:

beaches, natural sights, museums, shopping malls and cafeterias.

and the rest represent places shown during the night:

nightclubs, bars and restaurants.

Table 1 shows the query parameters used to gather cafeteria related places in Malta. These categories are based on the ones used by Wörndl et al. [6] for their application.

In return, the API returns a list of places of the specified area and category and attributes about each place. The attributes used by our application include the place's name, rating, the number of reviews and the coordinates.

3.2.3 Optimisation Algorithms

The following will discuss the steps required to produce the two timetable optimisation algorithms. PSO and GAs are two meta-heuristics that use a population to converge to a fit solution. Therefore, they require an initial random generation of possible timetables. In our algorithm, we introduce a method of randomisation bias which will be discussed before the algorithms.

Random Bias With this technique, the randomness of the initial population is weighted based on two components.

1. the place's rating and
2. the place's number of ratings.

Before the algorithm starts each POI will be given a score that will determine its probability of being chosen as part of the initial population. For example, if a POI i has a rating, r_i which a value from 0 to 5, and the number of ratings from users z_i. The score of i is calculated using the following equation:

$$\text{score} = \frac{r_i}{5} + \frac{z_i}{\max(\sum_{j=0}^{N} z_j)}$$

This bias gives a head start to the algorithm rather than just starting optimising from purely random itineraries, highly likely to be of bad quality.

Algorithm 1: Particle Swarm Optimisation

```
iter = x ;
count = 0 ;
particles = RandomBiasInitialiser(); ;
bestParticle = particle[0] ;
while count > x do
    i = 0 ;
    maxScore = 0 ;
    while i <= particles.length do
        /* Updating personal best                                    */
        if particles[i].score > particles[i].personalBest.score then
            particles[i].personalBest = particles[i].position;
        /* Updating global best                                      */
        if particles[i].score > maxScore then
            maxScore = particles[i].score ;
            bestParticle = particles[i].position ;
        i = i +1 ;
        /* Calculate new position                                    */
        particle[i].calculateNewPosition() ;
    return bestParticle
```

Particle Swarm Optimisation In PSO, the whole population is referred to as the swarm, whilst a single member a particle. Each particle has a 2-dimensional position(P) vector representing the current timetable solution and a 2-dimensional velocity(V) vector expressing the direction of the particle during its search period.

The algorithm has six integer parameters including the number of particles and the number of iterations. The personal acceleration (PA) affects how far away the particle

moves from the personal best position (PB). The global best acceleration (GA) attracts the global best position (GB) of the whole swarm. The inertia (I) constant helps the particle explore new solutions and escape the local minima through randomness.

At each iteration, the new velocity is calculated using

$$\text{new velocity} = \text{I} + (\text{PA} * (\text{PB} - \text{P})) + (\text{GA} * (\text{GB} - \text{P}))$$

the new position is calculated using

$$\text{new position} = \text{P} + \text{new velocity}$$

After a few iterations have passed, particles use their velocity and move towards the optimum position. We demonstrate the framework of our PSO algorithm in algorithm 1.

Genetic Algorithms Genetics algorithms use biological terms to describe their attributes. For example, a timetable solution in population is referred to as a chromosome. (cite)

In PSO, the algorithm optimises by allowing each particle to move closer to the global best every iteration. In comparison, in GAs, first, the best chromosomes known as the elites are selected from each iteration. Then, three techniques, namely selection, mutation, and crossover, are applied to generate the next population.

We used the geneticalgorithm2 package provided by X, which allowed us to use the same score function and the random bias to initialise the particles. The algorithm has X parameters. The parents portion represents the number of parents who will reproduce and create the next generation. The mutation probability determines the chance a POI in a chromosome will be replaced by a random value to which the algorithm will converge less quickly and explore more of the search space. The crossover probability will affect the chance that part of its solution goes to the child. Finally, the elite ratio determines how much of the best chromosomes in an iteration make it to the next iteration. There are many types of crossovers and mutations. In this algorithm, we explored X. The algorithm produced the following steps:

Step 1: Initialise the first population using random bias.
Step 2: Select the best chromosomes from the population.
Step 3: Select the elite particles that will make it to the next iteration.
Step 3: Apply Crossover, Mutation and Selection on the population.
Step 4: Check if the number of iterations has exceeded.

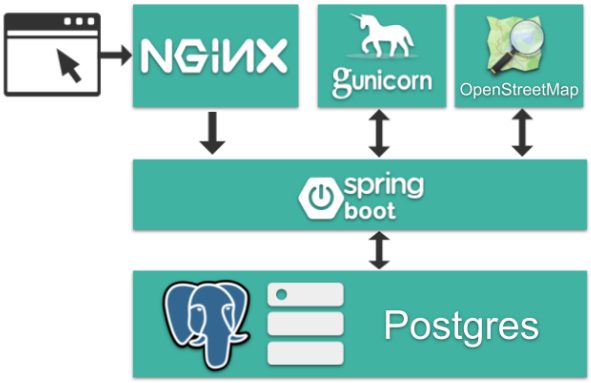

Figure 8: Tech Stack implementation of the application

3.3 In-Depth Semi-Structured Interviews

We asked 20 participants to use our application and asked them several questions to gather information about our user profiling technique and timetable generator. First, the application generated two timetables. A personalised itinerary and an itinerary where the travel interest vector is ignored; therefore, all POIs have an equal chance of being selected. However, the participants were not told which one was which, and we mixed the order of timetables to avoid anchor bias.

We first started discussed with the interviewee looks for in a holiday and the types of activities, so we could check whether this matches their travel interest vector. Then, after the timetables are generated, we asked them to review the results and which result they think is more personalised. At this stage, the interviewees were shown their automatically generated characteristics and asked what they would change, followed by whether they think their social media presence represents their travel preferences.

Finally, we discussed some considerations and future improvements regarding automatic preference gathering from social media and how this approach differs from their usual travel planning.

3.4 Web application implementation and user interface

We built the application using several technologies where each communicates with each other to provide a user-friendly website for the potential tourist. Figure 8 shows the tech stack diagram of the website. The website is accessible through the URL `https://www.touristplanner.xyz`. We built the front end of the website using HTML, CSS and

javascript and hosted it on a cloud Vultr server. The website is responsive to be accessible from both a mobile phone and a laptop. The website communicates with the back end of the application using REST endpoints, hosted on a separate dedicated server provided and developed the Java Spring Boot framework. Another Python Gunicorn server is used to generate the itinerary and calculate a travel interest vector which sends the information directly to the Spring boot server. Finally, a local instance of an Open Source Routing Machine server calculates the distances from one tourist attraction to another used by the Gunicorn server to optimise the itinerary.

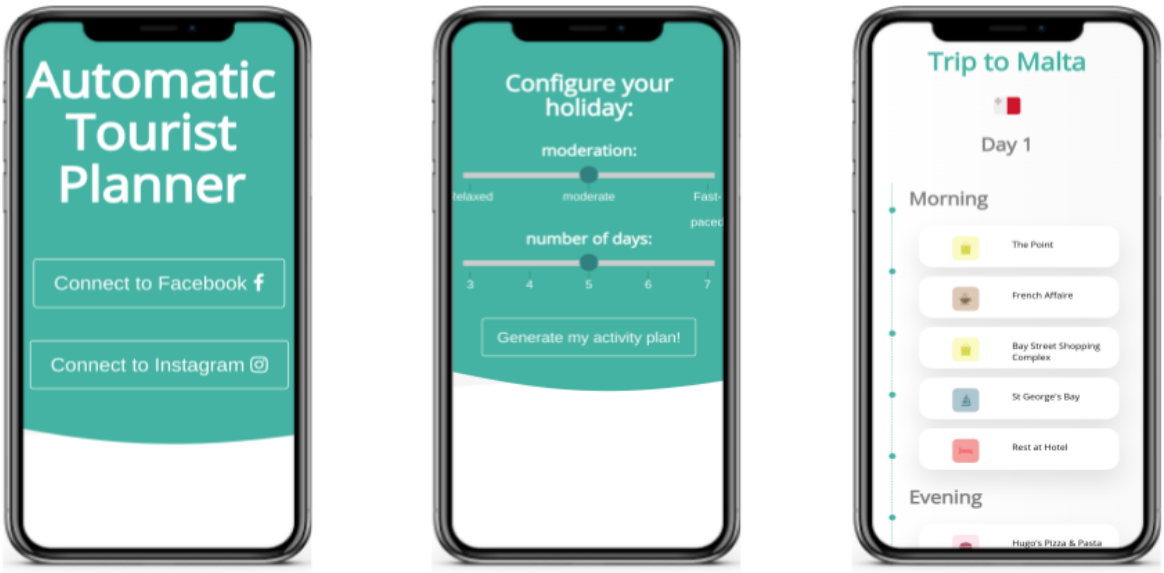

Figure 9: User Experience Timeline

Figure 9 shows screenshots of the website portraying the user's timeline. The user navigates to the homepage, accepts terms and conditions and connect his social media profiles. The user selects the number of days M and the activity activity pace C. The website navigates to the final page of the application exhibiting their personalised itinerary.

4 Results and Evaluation

This section will evaluate the image classification technique for the automatic user profiling and the itinerary generation algorithm separately. We will then assess the performance of the whole application and its effects through the interviews.

4.1 Automatic User Profiling

The following evaluates the performance results of the three CNNs used to classify the users' social media images.

Training results from the TensorFlow Keras sequential model We trained this model using the NVIDIA Tesla K80 GPU provided by Google Colab. At 16 epochs, the model starts to overfit as the validation loss starts to increase, as shown in figure 10. This is why we chose to calculate the model's performance on the testing set in the following subsections using the first 16 epochs.

Figure 10: Training and validation accuracy of the model on the testing and validation dataset

Performance comparison of the three models The testing dataset consists of 500 images gathered using the Unsplash API. We calculated the Accuracy, Precision, Recall and F1 Score on the testing set to evaluate the models by calculating the number of true positive (TP), true negative (TN), false-negative (FN) and false positive (FP) as a percentage over the dataset.

The accuracy of the models, shown in figure 11, is represented by:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

This shows how many category predictions a model got right. In this case, the Resnet 50, Resnet 18, and Keras models have an average 93.7%, 92.4%, and 88.7% average accuracy, respectively. Overall, the models have high accuracy. Therefore, we need to rely on other evaluation techniques.

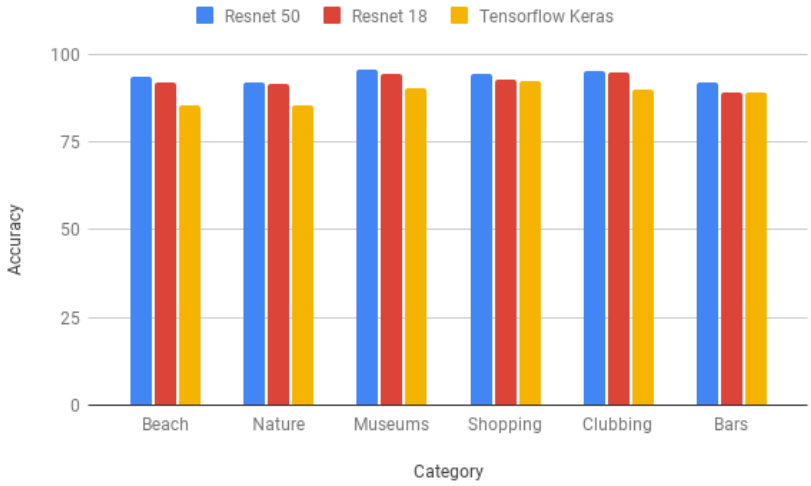

Figure 11: Accuracy of the models

The precision of the models, shown in figure 12, represented by

$$Precision = \frac{TP}{TP + FP}$$

This represents the ratio of correct predictions over the total positive labels. In this case, the Resnet 50, Resnet 18, and Keras models have an average 72.4%, 66.7%, and 62.4% precision rate, respectively. The Resnet 50 has the best average overall; however, the Keras model performed better for the 'clubbing' and 'bars' categories but very poorly in the shopping category.

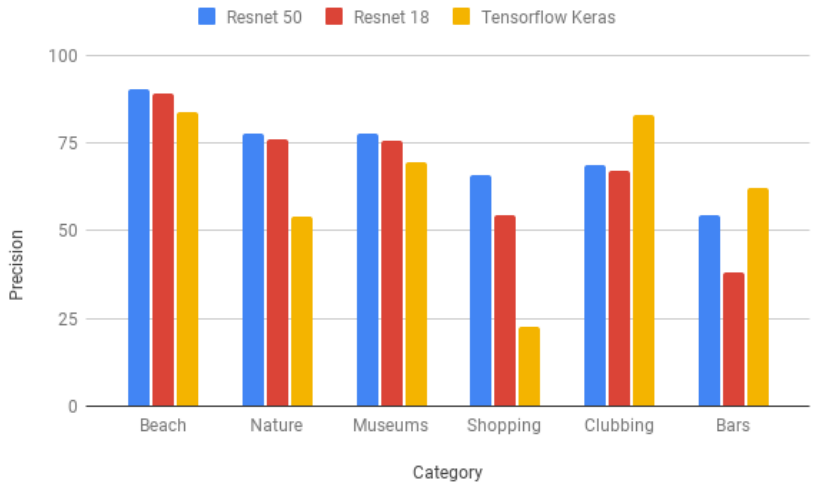

Figure 12: Precision of the models

The proportion of actual positive labels that the models identify correctly is represented by the recall value calculated using

$$Recall = \frac{TP}{TP + FN}$$

In this case, figure 13 shows how for the clubbing and bar categories, although the Resnet models were less accurate and less precise, they rarely labelled an image as a bar or a club when they are not supposed to.

Figure 13: Recall of the Resnet-18, Resnet-50 and Tensorflow Keras Sequential

The F1 score is the weighted average of Precision and Recall measured using:

$$F1Score = \frac{2 * Recall * Precision}{Recall + Precision}$$

The Reset 50 has a pretty consistent and high score on the testing dataset with an average F1 score of 76.29., so we decided to use it as our baseline for the tourist itinerary generation application.

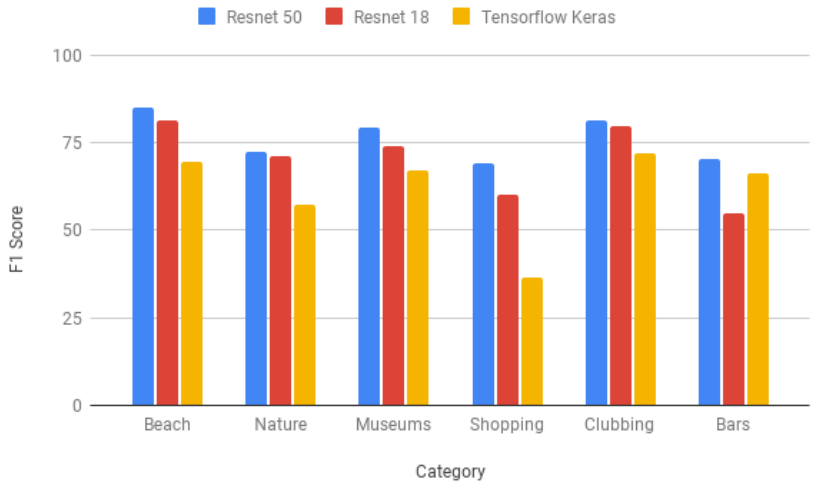

Figure 14: F1 score of the Resnet-18, Resnet-50 and Tensorflow Keras Sequential

These results have shown how a CNN could classify photos into the six application characteristics. Although all of the models achieved valid results, the Resnet 50 model was the best performer. Therefore, we will be using this model for our application.

4.2 Itinerary Optimisation Algorithms

We tested out the itinerary generation algorithms using 200 POIs in Malta. Figure 15 shows the score of 10 PSO itineraries increasing throughout several iterations for the morning and evening section of the day.

At around 12 iterations in figure 15, the algorithm generally converges. Figure X shows the spread of particles converging.

Increasing the number of particles also increases the average score, as shown in figure 16 for 10 generated itineraries and 14 iterations. However, the difference in average score between 50 and 100 particles is not proportional to the average time taken to create an

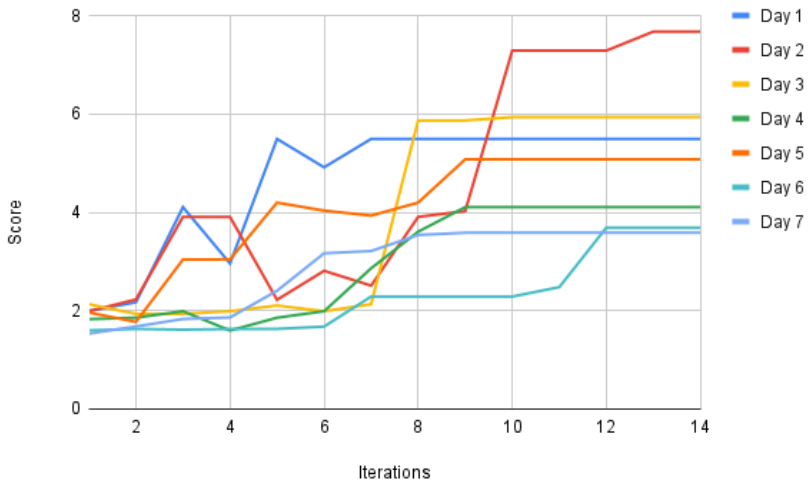

Figure 15: Training and validation accuracy of the model on the testing and validation dataset

itinerary for a single day in figure 17. That is why for the application, the population is set to 50.

The original algorithm X did not contain the inertia property; however, it was introduced by X since it controls the convergence behaviour. Out algorithm implements the Time-Varying Inertia Weight X, which gradually decreases throughout each iteration. Figure X shows the optimisation algorithm without the inertia property and barely increases the score since the particles do not explore new territories.

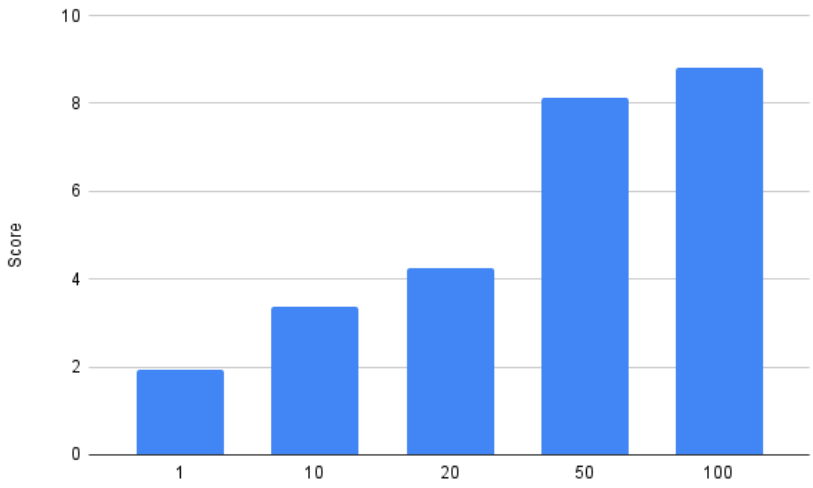

Figure 16: Training and validation accuracy of the model on the testing and validation dataset

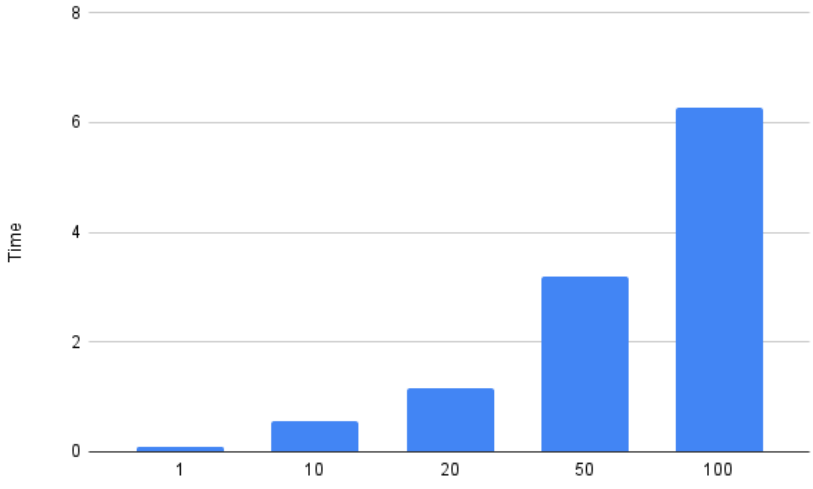

Figure 17: Training and validation accuracy of the model on the testing and validation dataset

References

[1] W. T. WTTC, "Travel & Tourism: Global Economic Impact & Issues 2018," 2018.

[2] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu, "Automatic construction of travel itineraries using social breadcrumbs," in *HT'10 - Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, 2010, pp. 35–44. [Online]. Available: http://www.munmund.net/pubs/ht{_}10{_}long.pdf

[3] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," pp. 1–10, feb 2011.

[4] A. Hoppe, A. Roxin, and C. Nicolle, "Semantic User Profiling for Digital Advertising," *Economic computation and economic cybernetics studies and research / Academy of Economic Studies*, vol. 49, 2015.

[5] A. Cufoglu, "User Profiling-A Short Review," Tech. Rep. 3.

[6] W. Wörndl, A. Hefele, and D. Herzog, "Recommending a sequence of interesting places for tourist trips," *Information Technology and Tourism*, vol. 17, no. 1, pp. 31–54, mar 2017. [Online]. Available: http://link.springer.com/10.1007/s40558-017-0076-5

[7] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 375–406, feb 2018. [Online]. Available: https://doi.org/10.1007/s10115-017-1056-y

[8] G. Tumas and F. Ricci, "Personalized Mobile City Transport Advisory System," in *Information and Communication Technologies in Tourism 2009*. Springer Vienna, 2009, pp. 173–183.

[9] D. Gavalas, V. Kasapakis, C. Konstantopoulos, G. Pantziou, N. Vathis, and C. Zaroliagis, "The eCOMPASS multimodal tourist tour planner Keywords: Tourist Trip Design Problem Multimodal tour planning Urban transportation Orienteering Problem Time window Time dependent travel time Context awareness Web service Mobile application," 2015. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2015.05.046

[10] T. N. Nguyen and F. Ricci, "A chat-based group recommender system for tourism," *Information Technology and Tourism*, vol. 18, no. 1-4, pp. 5–28, apr 2018. [Online]. Available: https://doi.org/10.1007/s40558-017-0099-y

[11] A. Terttunen, "The influence of Instagram on consumers' travel plan-ning and destination choice," Tech. Rep., 2017. [Online]. Available: http://www.theseus.fi/handle/10024/129932

[12] S. C. Guntuku, W. Lin, J. Carpenter, W. K. Ng, L. H. Ungar, and D. Preotiuc-Pietro, "Studying personality through the content of posted and liked images on Twitter," in *WebSci 2017 - Proceedings of the 2017 ACM Web Science Conference*. Association for Computing Machinery, Inc, jun 2017, pp. 223–227.

[13] M. Chen, A. Zheng, and K. Q. Weinberger, "Fast Image Tagging," Tech. Rep., 2013. [Online]. Available: http://tinyurl.com/9jfs7ut

[14] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems," *Journal of Heuristics*, vol. 20, no. 3, pp. 291–328, jun 2014. [Online]. Available: http://link.springer.com/10.1007/s10732-014-9242-5

[15] I. Memon, L. Chen, A. Majid, M. Lv, I. Hussain, and G. Chen, "Travel Recommendation Using Geo-tagged Photos in Social Media for Tourist," vol. 80, pp. 1347–1362, 2015.

[16] C. Lucchese, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini, "How Random Walks can Help Tourism," 2012. [Online]. Available: http://www.flickr.com

[17] K. Hui Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized Tour Recommendation based on User Interests and Points of Interest Visit Durations," Tech. Rep., 2015.

[18] K. Hui Lim, S. Karunasekera, C. Leckie, and J. Chan, "Recommending Tours and Places-of-Interest based on User Interests from Geo-tagged Photos." [Online]. Available: http://dx.doi.org/10.1145/2744680.2744693.

[19] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, "Travel route recommendation using geotagged photos," *Knowledge and Information Systems*, vol. 37, no. 1,

pp. 37–60, oct 2013. [Online]. Available: https://link.springer.com/article/10.1007/s10115-012-0580-z

[20] T. Kurashima and K. Fujimura, *Travel Route Recommendation Using Geotags in Photo Sharing Sites*, 2010.

[21] I. Brilhante, J. Antonio Macedo, F. Maria Nardini, R. Perego, and C. Renso, "Where Shall We Go Today? Planning Touristic Tours with TripBuilder," 2013. [Online]. Available: http://dx.doi.org/10.1145/2505515.2505643.

[22] I. R. Brilhante, J. A. Macedo, F. M. Nardini, R. Perego, and C. Renso, "On planning sightseeing tours with TripBuilder," *Information Processing and Management*, vol. 51, no. 2, pp. 1–15, mar 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0306457314000922

[23] Y. Zheng and X. Xie, "Learning Travel Recommendations from User-Generated GPS Traces," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 1, 2011. [Online]. Available: https://doi.org/10.1145/1889681.1889683

[24] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining Interesting Locations and Travel Sequences from GPS Trajectories," in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 791–800. [Online]. Available: https://doi.org/10.1145/1526709.1526816

[25] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *Proceedings - International Conference on Data Engineering*, 2011, pp. 900–911.

[26] K. H. Lim, J. Chan, S. Karunasekera, and C. Leckie, "Tour recommendation and trip planning using location-based social media: a survey," *Knowledge and Information Systems*, vol. 60, no. 3, pp. 1247–1275, dec 2018. [Online]. Available: https://link.springer.com/article/10.1007/s10115-018-1297-4

[27] T. Tsiligirides, "Heuristic methods applied to orienteering," *Journal of the Operational Research Society*, vol. 35, no. 9, pp. 797–809, sep 1984. [Online]. Available: https://www.tandfonline.com/doi/full/10.1057/jors.1984.162

[28] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," pp. 1–10, feb 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0377221710002973

[29] G. Kobeaga, M. Merino, and J. A. Lozano, "An efficient evolutionary algorithm for the orienteering problem," *Computers and Operations Research*, vol. 90, pp. 42–59, feb 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0305054817302241

[30] X. Wang, B. L. Golden, and E. A. Wasil, "Using a genetic algorithm to solve the generalized orienteering problem," *Operations Research/ Computer Science Interfaces Series*, vol. 43, pp. 263–273, 2008.

[31] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering Problem: A survey of recent variants, solution approaches and applications," pp. 315–332, dec 2016.

[32] V. F. Yu, P. A. A. N. Redi, P. Jewpanya, A. Gunawan, V. F. . Yu, P. A. A. N. . Redi, P. . Jewpanya, V. F. Yua, A. A. N. Perwira Redia, and P. Jewpanyaa, "Selective discrete particle swarm optimization for the team Selective discrete particle swarm optimization for the team orienteering problem with time windows and partial scores orienteering problem with time windows and partial scores Citation Citation S," Tech. Rep., 2019. [Online]. Available: https://ink.library.smu.edu.sg/sis{_}research/4469

[33] A. Rezaee Jordehi and J. Jasni, "Parameter selection in particle swarm optimisation: A survey," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 25, no. 4, pp. 527–542, dec 2013. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/0952813X.2013.782348

[34] A. Z. Şevkli and F. E. Sevilgen, "StPSO: Strengthened particle swarm optimization," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 18, no. 6, pp. 1095–1114, nov 2010.

[35] Z. Sevkli and F. E. Sevilgen, "Discrete particle swarm optimization for the orienteering problem," in *2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010*. IEEE, jul 2010, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/5586532/

[36] A. Santini, "An adaptive large neighbourhood search algorithm for the orienteering problem," *Expert Systems with Applications*, vol. 123, pp. 154–167, jun 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417418308182

[37] M. G. Kantor and M. B. Rosenwein, "The orienteering problem with time windows," *Journal of the Operational Research Society*, vol. 43, no. 6, pp. 629–635, 1992.

[38] D. A. Herzog, "A User-Centered Approach to Solving the Tourist Trip Design Problem for Individuals and Groups," Tech. Rep., 2020.

[39] F. V. Fomin and A. Lingas, "Approximation algorithms for time-dependent orienteering," *Information Processing Letters*, vol. 83, no. 2, pp. 57–62, jul 2002.

[40] R. A. Abbaspour and F. Samadzadegan, "Time-dependent personal tour planning and scheduling in metropolises," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12 439–12 452, sep 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417411005410

[41] I. M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 464–474, feb 1996.

[42] K. Sylejmani, J. Dorn, and N. Musliu, "Planning the trip itinerary for tourist groups," *Information Technology and Tourism*, vol. 17, no. 3, pp. 275–314, sep 2017. [Online]. Available: https://link.springer.com/article/10.1007/s40558-017-0080-9

[43] S. Muthuswamy and S. S. Lam, "Discrete particle swarm optimization for the team orienteering problem," *Memetic Computing*, vol. 3, no. 4, pp. 287–303, dec 2011. [Online]. Available: http://link.springer.com/10.1007/s12293-011-0071-x

[44] W. Wisittipanich and C. Boonya, "Multi-objective Tourist Trip Design Problem in Chiang Mai City," in *IOP Conference Series: Materials Science and Engineering*, vol. 895, no. 1. Institute of Physics Publishing, jul 2020, p. 012014. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/895/1/012014https://iopscience.iop.org/article/10.1088/1757-899X/895/1/012014/meta

[45] K. Sylejmani, J. Dorn, and N. Musliu, "A Tabu Search approach for Multi Constrained Team Orienteering Problem and its application in touristic trip planning," in *Proceedings of the 2012 12th International Conference on Hybrid Intelligent Systems, HIS 2012*, 2012, pp. 300–305.

[46] D. Miller, J. Sinanan, X. Wang, T. McDonald, N. Haynes, E. Costa, J. Spyer, S. Venkatraman, and R. Nicolescu, *How the World Changed Social Media*. UCL Press, feb 2016. [Online]. Available: www.ucl.ac.uk/ucl-press

[47] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 Million Image Database for Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, jun 2018.

[48] "Geo-location APIs — Google Maps Platform — Google Cloud." [Online]. Available: https://cloud.google.com/maps-platform

[49] H. Iltifat, "Generation of paths through discovered places based on a recommender system," Ph.D. dissertation, Master's Thesis, Department of Computer Science, Technical University of˜. . . , 2014.