



Deep Learning-based Mobile Tourism Recommender System

Dhomas Hatta Fudholi^{1*}, Septia Rani², Dimastyo Muhaimin Arifin³, Mochamad Rezky Satyatama⁴

^{1,2,3,4}Department of Informatics, Universitas Islam Indonesia, Indonesia

Abstract

Purpose: This study developed a deep learning-based mobile travel recommendation system that provides recommendations for local tourist destinations based on users' favorite travel photos. To provide recommendations, use cosine similarity to measure the similarity score between a person's image and a tourism destination gallery through the tag label vector. Label tags are inferred using an image classifier model run from a mobile user device via TensorFlow Lite. There are 40 tag labels that refer to categories, activities and objects of local tourism destinations.

Methods: The model is trained using state-of-the-art mobile deep learning architecture EfficientNet-Lite, which is new in the domain of tourism recommender system.

Result: This research has conducted several experiments and obtained an average model accuracy of more than 85%, using EfficientNet-Lite as its basic architecture. The implementation of the system as an Android application is proven to provide excellent recommendations with a Mean Absolute Percentage Error (MAPE) of 5.2%.

Novelty: A tourism recommendation system is a crucial solution to help tourists discover more diverse tourism destinations. A content-based approach in a recommender system can be an effective way of recommending items because it looks at the user's preference histories. For a cold-start problem in the tourism domain, where rating data or past access may not be found, we can treat the user's past-travel-photos as the histories data. Besides, the use of photos as an input makes the user experience seamless and more effortless.

Keywords: Deep Learning, EfficientNet-Lite, Image, Recommender System, Tourism

Received February 2021 / **Revised** May 2021 / **Accepted** May 2021

This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/)



INTRODUCTION

One of the critical steps that must be taken when someone decides to go on a tour is determining the tourist attractions to be visited. Each city has its characteristics, and it is often different from each other, both in terms of types of tourist attractions and the number of tourist attractions provided. For example, in Paris (France), a search on Tripadvisor produced 3096 search results for tourist attractions. These many choices are certainly an advantage for tourists, but on the other hand, it becomes a challenge in which tourists must determine exciting places that are in line with their preferences. Choosing a tourism destination becomes more complicated when the choice of tourist destinations in a city is vast, but the tourist's visit-time is limited. In this case, a tourism recommendation system can be a solution to help tourists make the right decisions and save time.

The recommendation system can be implemented using several methods. Some of the well-known recommender systems methods include the collaborative filtering method, content-based method, and knowledge-based method [1]. In collaborative filtering, it usually uses a neighborhood model that involves several users, assuming that similar user profiles will also have similar preferences. In the knowledge-based method, users interactively state their specifications, and then the system will combine them with domain knowledge to produce recommendations. Meanwhile, the content-based approach uses the basic idea that the user will be interested in items that he/she has rated or accessed in the past. This content-based method is one of the effective ways of recommending items because it looks at the user's preference history. However, for a cold-start problem in the recommender system domain, the existence of rating data or past

* Corresponding author.

Email addresses: hatta.fudholi@uii.ac.id (Fudholi), dimastyo.arifin@students.uii.ac.id (Arifin), septia@uii.ac.id (Rani), mochamad.satyatama@students.uii.ac.id (Satyatama)

DOI:10.15294/sji.v8i1.29262

access may not be found. In the case of tourism recommendations, user history data can be replaced with the user's past-travel-photos. The use of photos input to get a recommendation can also make it easier for users because users don't need to explain their text preferences.

Recommender systems for tourist attractions have drawn considerable attention from researchers. A tourist attractions recommender system that uses check-in data on social media (such as Facebook, Twitter, Foursquare) could give personalized tourist attractions recommendations [2]. This approach can resolve the cold-start problem in the recommender system. Another strategy is using collective intelligence, which is gathered from a large amount of user-generated content in social media such as Flickr, Tripadvisor, and ¹hatta.fudholi@uii.ac.id Wikitravel [3]. Different companies have exploited deep learning methods to improve their recommender system's diversity and performance [4], [5]. For instance, [4] introduced a Google Play recommendation that uses broad and deep architecture. [6] proposed a movie recommendation system using a deep learning-based method. [5] utilized RNN-based architecture for news recommendations. [7] proposed a novel content-based approach based on probabilistic graphical model and the deep belief network. Deep learning based recommendation has proven its capability to enhance performance of the existing types of recommendation system [8].

In recent years, mobile phones' growing use appeals to researchers to pursue the implementation of recommendation systems in mobile applications [9]. [10] proposed a combination of user preferences and their location to generate recommendations through a hybrid recommender approach. [11] used an ontology-based approach to generate museum websites' recommendations along with its description by utilizing the user's query and location. Another form of mobile recommendation is implemented by [12]. They combined knowledge-based recommender and collaborative filtering to recommend user routes between two points. To overcome the limitations of GPS and RFID, [13] proposed a new approach to use a user's location to find a recommendation by using received signal strength (RSS) transmitted by their mobile phone. The captured RSS, combined with user's preferences, has proven its effectiveness in generating recommendations and attracting more users to use their recommender system. [14] proposed movie showtime with the context-aware knowledge-based recommender. In their research, the crowd, time, and user's location were taken into account to generate recommendations. [15] proposed a mobile recommender to assist taxi driver by recommending pick up points.

In this study, a tourism recommendation system that uses input in the form of images is designed. The photo images used can be in the form of a user's travel history photos or photos of tourist objects and activities that the user likes, which is referred to as a photo query. The deep learning technique is used to classify photos. Then we measure the similarities between the query photos with the photos in the database. As a case study, we use photographs of tourist objects and tourist activities in Yogyakarta (Indonesia). With this system, it is expected that it can improve the accuracy of the information provided to tourists, especially in selecting tourist attractions that match their preferences.

The main contribution of this paper is that we implemented our deep learning recommendation model using the edge architecture EfficientNet-Lite, which is derived from the state-of-the-art architecture EfficientNet. EfficientNet architecture is 8.4x smaller and 6.1x faster on inference than the best existing CNN when testing on the ImageNet dataset [16]. Although relatively new, EfficientNet has been implemented in several cases such as for image recognition [17], [18], speech recognition [19], and also detection in videos [20]. In general, EfficientNet improves overall prediction/recognition accuracy. According to [17] EfficientNet is also suitable to be deployed onto devices with limited computational resources, such as in mobile phones. Thus, in this research, we employ EfficientNet-Lite architecture in our tourism recommender system.

METHODS

The research is conducted through three main phases: data collection, modeling, and implementation. There are two different kinds of data collected in this research. The first collection of data is labeled images that become the dataset in the developing classifier model. The second collection of data is local tourism destination images, used as the base in the recommendation process. This research created two models: (i) image classifier model that classifies images into the local tourism-related category, and (ii) recommendation model that consists of a matching algorithm that we design and use to give a recommendation. Finally, the model is implemented in a mobile-based application and evaluated.




Data Collection

We have defined 40 different classes used in the image classifier, as seen in Table 1. The 40 classes represent the tourism domain and are classified into three attributes: category, activity, and object. In the category attribute, we have 24 classes and focused on collecting images of tourist destinations based on its types, such as beach, mountain, and temple. The activity attribute reflects 11 specific tourist activities that can be done in the tourism destination, such as climbing and swimming. The object attribute consists of local specific tourism-related objects, such as gamelan (traditional musical instrument) and batik (traditional fashion motives). Our study included classes derived from the knowledge base taken from local tourism authorities' sources. As our case study, we take a local tourism domain in Yogyakarta, Indonesia. Hence, we put the five items within the object attribute listed in Table 1. We collected relevant images from the web using the Google image search engine. The sample image dataset can be seen in Table 2.

Table 1. Classes of data

Attribute	Values
category	(1) reservoir, (2) hill, (3) lake, (4) zoo, (5) museum, (6) garden, (7) desert, (8) tomb, (9) waterboom, (10) waterfall, (11) valley, (12) beach, (13) park, (14) mountain, (15) cave, (16) river, (17) historical place, (18) monument, (19) art gallery, (20) temple, (21) amusement park, (22) library, (23) culinary, (24) mall
activity	(25) climbing, (26) swimming, (27) cave exploring, (28) jogging, (29) camping, (30) gardening, (31) fishing, (32) boat riding, (33) painting, (34) sculpting, (35) taking pictures
object	(36) bakpia, (37) puppet, (38) batik, (39) gamelan, (40) becak

Table 2. Sample image dataset

Attribute	Category	Activity	Object
Value	mountain	camping	puppet
Sample Image			

Modeling: Classifier

In this research, we experimented with two different CNN based pretrained models. We employed EfficientNet model as the base. We will train the five different variants of EfficientNet-Lite models with our dataset and the model with the best performance will be selected as our baseline model to generate recommendation. Especially for the newer state-of-the-art architecture, EfficientNet, the baseline CNN can be scaled in three dimensions: width, depth, and resolution. The width dimension indicates how much neurons are located in a layer. The depth dimension corresponds to the number of layers in a neural network. Finally, the resolution dimension indicates the width and height of the image. The conventional scaling method usually scales one of the three dimensions. EfficientNet managed to scale CNN by scaling these three dimensions, referred to as compound scaling [16]. Hence, in this research, we use EfficientNet as the base architecture to model the image classifier.

Although EfficientNet has proved its capability to predict better using scaled CNN [16], its performance while performing inference on mobile devices is dropped. Inference time took longer compared to other models. These drawbacks may be inconvenient for users. Therefore, Google introduced EfficientNet-Lite to address this issue. They replaced all swish activations with RELU-6 and fixed the stem and head while scaling models to reduce size. There are five different EfficientNet-Lite architectures (EfficientNet_lite_0, EfficientNet_lite_1, EfficientNet_lite_2, EfficientNet_lite_3, EfficientNet_lite_4) where the number of conv2d layer to conv2d RELU6 layer iterations is different, for instance 13 iterations for EfficientNet_lite_3 and 8 iterations for EfficientNet_lite_0. The higher the number of the architecture name, the more complex and bigger the architecture. In this research, we used all five EfficientNet-Lite architectures in our experiment and analyzed the result.

Modelling: Recommender

To generate recommendations, we used users' images in a tourist attraction or their activity at that attraction as the input. Then we use the model to predict the probability for every 40 classes for each image. The predicted 40 probabilities are then stored into a vector. Especially for the tourism destination images, we apply a threshold of 0.3 in the classification probability vector. When the tourism destination's prediction value is higher than 0.3, it becomes 1, otherwise 0. Giving value conversion gives a definite category of tourism destinations. Figure 1 shows the sample tourism destination image that may be inferred by our model. On the same figure, the vector besides the image is the vector constructed after the inference process, which is arranged based on labels included in Table 1, respectively. The 1 value within the vector are for the lake, waterfall, river, and swimming respectively. In some conditions, users may select multiple input images. Hence, to predict probability under this condition, we take the maximum value of all images' possibilities. The newly generated vector, called a query in this study, will be compared with the vectors of tourist attractions stored in the database to calculate the similarity value.

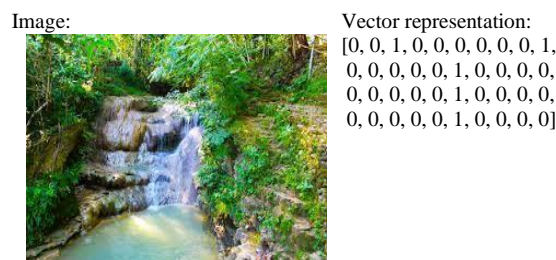


Figure 1. Example of vector representation of the tourism destination image

To give a better view of our recommendation process, we created a flowchart (shown in Figure 2) that illustrates our method to generate a recommendation. In general, the system will convert the input images to their vector representation using the image classifier model given. The vector becomes a query to be matched with our tourism destinations database using a similarity equation. To measure similarity, we employed the Cosine Similarity method as shown in equation (1), where a is the query vector, and b is the tourism destination vector. The output of the recommendation will be a tourism destination that has a similarity score greater than 0.5.

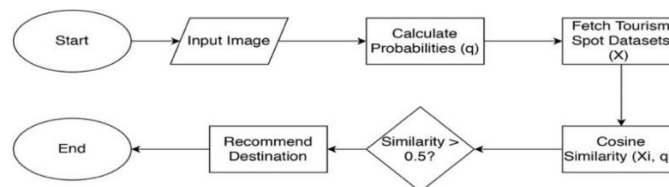


Figure 2. Recommendation process

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|} \quad (1)$$

Implementation and Evaluation

To complete the phase on building the recommendation system, we implemented our model into an Android application, using MySQL as the database. Tensorflow Lite framework is also used. The evaluations we performed in this study are classifier model performance evaluation and recommendation evaluation. We evaluate models' accuracy and loss in all training, validation, and testing dataset to measure our model performance. Moreover, we also evaluate the models based on their inference time and size of input images.

Recommendation model evaluation will involve 10 users who frequently travels (about 6-7 times in a year). Each user will be asked to select images of them traveling and upload it to our application. Users are asked to create their own similarity score of each attraction that are recommended for them. These similarity score will be subtracted with the similarity score generated by the application and will be called as error. The evaluation method chosen in this research are Mean Absolute Error (MAE), shown on Equation 2.

We also calculate Mean Absolute Percentage Error (MAPE) using Equation 3. Since MAPE is percentage, it is more convenient to determine our recommendation's performance.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - y_i'| \quad (2)$$

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - y_i'}{y_i} \right| \quad (3)$$

Where y_i = user's similarity i -th value; y_i' = generated similarity i -th value; n = total amount of data

RESULT AND DISCUSSION

In our research, we utilized EfficientNet_lite_0 (EL-0), EfficientNet_lite_1 (EL-1), EfficientNet_lite_2 (EL-2), EfficientNet_lite_3 (EL-3), and EfficientNet_lite_4 (EL-4) architectures which was derived from EfficientNet-B0, EfficientNet-B1, EfficientNet-B2, EfficientNet-B3, and EfficientNet-B4 respectively. We set the number of epochs equal to 20 with a batch size of 32 for all models that we trained during the training process.

Table 3 shows the loss and the accuracy of our developed tourism-related image classifier model using five different architectures against the training dataset, the validation dataset, and the test dataset. During the training, EfficientNet trained models perform well, with the value of accuracy being in between 93%-96%. We can also see the average values of the accuracy between the six models are negligible.

Table 3. Lost and accuracy of the trained models

Model	Train Dataset		Validation Dataset		Test Dataset		Acc Average
	Loss	Acc	Loss	Acc	Loss	Acc	
EL-0	0.95	96%	1.31	80%	1.18	84%	87%
EL-1	0.98	95%	1.20	85%	1.18	83%	88%
EL-2	0.99	94%	1.00	87%	1.17	83%	88%
EL-3	0.96	95%	1.23	83%	1.16	85%	88%
EL-4	1.03	93%	1.24	81%	1.23	84%	86%

Figure 3 depicts more detailed loss and accuracy values during the training process. Figure 3(a) shows the accuracy values in each epoch against the training dataset. From the figure, we can see that all five models' accuracy rises in almost the same manner in the early epoch. Differently, Figure 3(b) shows the accuracy values in each epoch against the validation dataset. From the figure, we can see that EL-1, EL-2, and EL-3 accuracy values are quicker to raise at the beginning of the training iteration. However, once again, the difference in accuracy value between all models in each epoch is negligible. The small differences may be due to the number of classes (40) is still quite small. Hence, smaller architectures can still keep pace with higher architecture. From the experiments that we had done, we achieved a quite similar average accuracy for the five different architecture of EfficientNet_Lite with the highest accuracy against our training, validation, and testing dataset are 96% (EL-0), 87% (EL-2), and 85% (EL-3) respectively.

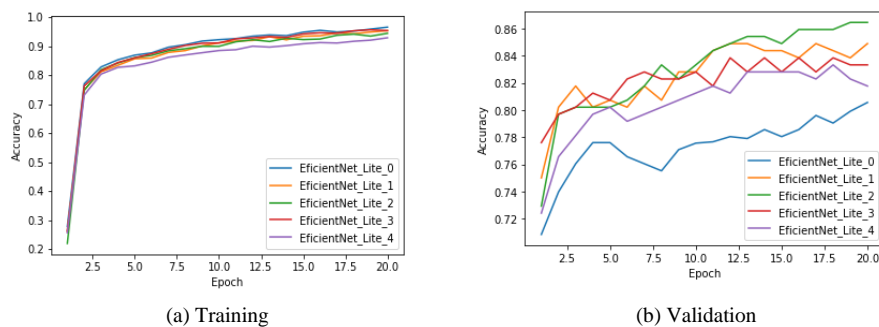


Figure 3. The accuracy values of the models

The five EfficientNet variants is employed to perform inference on mobile devices. The models are converted into a Tensorflow Lite format to be implemented in mobile applications. Table 4 shows the size of the exported model and the input size when performing the inferencing. The developed model size comparison is pretty obvious. EL-4, which has the biggest architecture (most extensive number of parameters), has the largest file size. EL-4 is also taking the longest time during the inference process. When the small accuracy difference is tolerable, the model with the smallest size is the best choice to be implemented in our case study.

Table 4. The size and inference time for each model

Model	Input Size	Size (MB)	Inference Time (s)
EL-0	224 x 224	13	0.096
EL-1	240 x 240	16	0.102
EL-2	260 x 260	19	0.145
EL-3	280 x 280	27	0.190
EL-4	300 x 300	45	0.312

Our research aims to create a system that recommends tourism destinations to users on the mobile application. Our research also aims to integrate Twitter with our mobile application, so we created a Rest API to scrap user's picture from Twitter in Python. Figure 4(a) is the first page in the recommendation part presented to the user. On this page, users are asked to fill their Twitter username which will be used as the part of our Twitter integration. Then, user has the option to select images from their Twitter account or from their phone gallery. After that, the application will begin to count the label tags' probability for the images through the assigned image classifier model. For instance, we have a sample of a user's photo depicted in Figure 4(b). The result of the classification process using the EL-3 model is climb (0.892289), taking pictures (0.027952), mountain (0.025383), jogging (0.015131), hill (0.013965). The classification result was then converted to its vector format and is compared to all the tourism destinations vector in the system database.

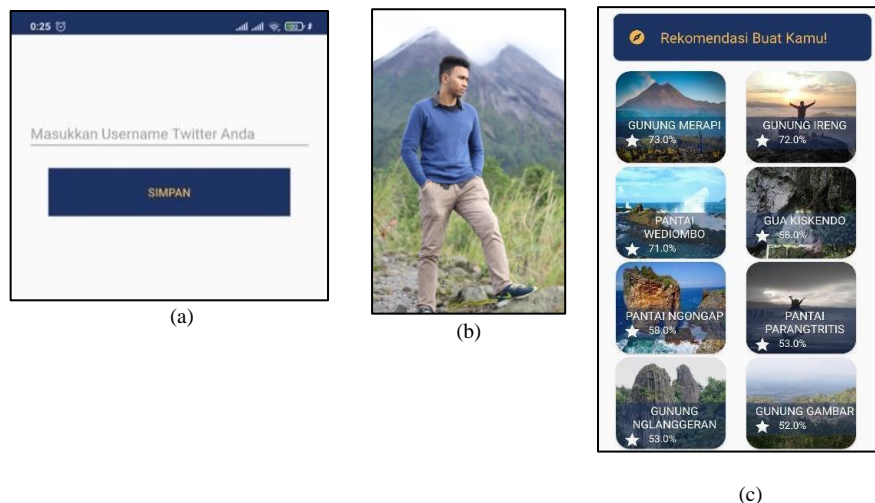


Figure 4. Application overview

Figure 4(c) shows the page that contains the generated tourist attractions. Each recommendation item consists of a picture, name, and similarity score. From the figure, we can see that the recommender system has been successfully given relevant recommendations to the user. Class label "climb" was the highest probability label predicted by our model. Hence, although we still get some "beach" as the recommended item, the topmost of the recommended destinations are mountain or climbing, with a high similarity percentage.

In this research, we gather 10 users to assess the recommendation model. We asked them to create their own similarity score for each place that are recommended for them. The result of our evaluation is shown at Table 5. Table 5 shows the various MAE and MAPE for 10 users with person E having the highest MAE and MAPE which may indicate that our recommendation does not suit well for her. Our model's average scores are 4.1 on MAE and 5% on MAPE. Based on MAPE score, our recommendation model can be considered as highly accurate. Another evaluation method conducted in this research is through matching score. We calculated the number of match and not match in regards to five given recommendations. Based on Table 5, person B and person E considered that the recommended attractions do not really suit their needs. However, most of the respondent (80%) gives more than 50% matching score. Therefore, our recommendation system can be considered as good recommender system.

Table 5. User assessment based on MAE, MAPE, Precision

User	MAE	MAPE	# of Match	# of Not Match	Precision
Person A	5.8	7%	4	1	80%
Person B	2.4	3%	1	4	20%
Person C	4.4	5%	3	2	60%
Person D	2.6	4%	3	2	60%
Person E	9.4	10%	1	4	20%
Person F	3.2	4%	4	1	80%
Person G	2.6	4%	4	1	80%
Person H	4.0	4%	3	2	60%
Person I	1.8	3%	4	1	80%
Person J	5.2	8%	4	1	80%
Average	4.1	5.2%	3.1	1.9	62%

CONCLUSION

Tourism destination recommendations through past-travel-photo can make the recommendation experience more seamless to the user. Through deep learning technology, we developed a mobile recommendation system. We take the state-of-the-art image classification architecture EfficientNet as the base, especially the edge architecture EfficientNet_Lite, to train image classifier models in the tourism domain. From the experiments that we had done, we achieved a quite similar average accuracy for the five different architecture of EfficientNet_Lite with the highest accuracy against our training, validation, and testing dataset are 96% (EL-0), 87% (EL-2), and 85% (EL-3) respectively. Also, the implementation of the model in the Android application can give good recommendations with MAE average score equals to 4.1 and MAPE average score equals to 5.2%.

REFERENCES

- [1] Charu C. Aggarwal, *Recommender Systems*, vol. 40, no. 3. 1997.
- [2] K. Kesorn, W. Juraphanthong, and A. Salaiwarakul, "Personalized Attraction Recommendation System for Tourists Through Check-In Data," *IEEE Access*, vol. 5, pp. 26703–26721, 2017.
- [3] J. Shen, C. Deng, and X. Gao, "Attraction recommendation: Towards personalized tourism via collective intelligence," *Neurocomputing*, vol. 173, pp. 789–798, 2016.
- [4] H. T. Cheng *et al.*, "Wide & deep learning for recommender systems," *ACM Int. Conf. Proceeding Ser.*, vol. 15-Septemb, pp. 7–10, 2016.
- [5] S. Okura, Y. Tagami, S. Ono, and A. Tajima, "Embedding-based News Recommendation for Millions of Users," *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.* vol. 25, pp. 388–392, 1946.
- [6] P. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," *Proc. 10th ACM Conf. Recomm. Syst.*, pp. 191–198, 2016.
- [7] X. Wan and Y. Wang, "Improving Content-based and Hybrid Music Recommendation using Deep Learning," *MM '14 Proc. 22nd ACM Int. Conf. Multimed.*, 2014.
- [8] A. Singhal, P. Sinha, and R. Pant, "Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works," *Int. J. Comput. Appl.*, vol. 180, no. 7, pp. 17–22, 2017.
- [9] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "Mobile recommender systems in tourism," *J. Netw. Comput. Appl.*, vol. 39, no. 1, pp. 319–333, 2014.
- [10] J. M. Noguera, M. J. Barranco, R. J. Segura, and L. Martínez, "A mobile 3D-GIS hybrid recommender system for tourism," *Inf. Sci. (Ny)*, vol. 215, pp. 37–52, 2012.
- [11] T. Ruotsalo *et al.*, "SMARTMUSEUM: A mobile recommender system for the Web of Data," *J. Web Semant.*, vol. 20, pp. 50–67, 2013.
- [12] D. Herzog, H. Massoud, and W. Wörndl, "Routeme: A mobile recommender system for personalized, multi-modal route planning," *UMAP 2017 - Proc. 25th Conf. User Model. Adapt. Pers.*, pp. 67–75, 2017.
- [13] B. Fang, S. Liao, K. Xu, H. Cheng, C. Zhu, and H. Chen, "A novel mobile recommender system for indoor shopping," *Expert Syst. Appl.*, vol. 39, no. 15, pp. 11992–12000, 2012.
- [14] L. O. Colombo-Mendoza, R. Valencia-García, A. Rodríguez-González, G. Alor-Hernández, and J. J. Samper-Zapater, "RecomMetz: A context-aware knowledge-based mobile recommender system for movie showtimes," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1202–1222, 2015.
- [15] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. J. Pazzani, "An Energy-Efficient

- Mobile Recommender System,” *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. data Min.*, 2010.
- [16] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019.
- [17] L. T. Duong, P. T. Nguyen, C. Di Sipio, and D. Di Ruscio, “Automated fruit recognition using EfficientNet and MixNet,” *Comput. Electron. Agric.*, vol. 171, no. April, 2020.
- [18] P. Zhang, L. Yang, and D. Li, “EfficientNet-B4-Ranger: A novel method for greenhouse cucumber disease recognition under natural complex environment,” *Comput. Electron. Agric.*, vol. 176, no. July, p. 105652, 2020.
- [19] X. Qidong, L., Yingying, L., Zhilian, Q., Xiaowei, L., & Yun, “Speech Recognition using EfficientNet,” *Proc. 2020 5th Int. Conf. Multimed. Syst. Signal Process.*, no. 159, pp. 64–68, 2020.
- [20] A. Noor, B. Benjdira, A. Ammar, and A. Koubaa, “DriftNet: Aggressive Driving Behavior Classification using 3D EfficientNet Architecture,” *Proc. ArXiv.*, 2020.