SnapShare: AI Trained Mobile App to Share Snaps Automatically

Ahmad Waqas*[1], Abdul Rehman Gilal[1], Adil Khan[1], Mazni Omar[2],
Murk Chohan[1], Ruqaya Gilal[2]

[1]Department of Computer Science, Sukkur IBA University, Pakistan
[2]School of Computing, Universiti Utara Malaysia
**[1]ahmad.waqas@iba-suk.edu.pk*

Abstract

These days people take more than 1 million group or selfie photos per day. This goes very hectic for a mobile owner to identify photos of each individual and send them their photos separately. Sharing photos create extra burden for mobile owners. There are fewer applications available (i.e., 23Snaps, Cluster, Path, letmesee) to share photos with small circle of friends. Unfortunately, these developed apps require user's interaction to identify individuals in the photo. This study proposes a SnapShare mobile application that uses Face Recognition Algorithms to classify individuals in the photos and automatically shares photos with recognized individuals. SnapShare basically uses Deep learning (DL) and Machine Learning (ML) techniques for Face Recognition from the captured images. Based on the results, the developed system achieves the standard performance accuracy (i.e., >90%). The aim of the SnapShare is to create comfort for mobile owners and people visible in-group photo to share and access photo automatically. Furthermore, SnapShare also facilitates user to back up their photo gallery on server storage.

Keywords: *SnapShare, Mobile App, Deep learning (DL), Machine Learning (ML) and Face Recognition.*

1. Introduction

People rarely miss any events (i.e., parties or outing with friends.) without having photo session [1]. It is observed that; photo session has become a mandatory part of each event. These day people take more than 1 million photos or group selfies per day. The captured photos could be of an individual all alone or in a group. There are multiple people in a group photo but the photo lies in the mobile that captured the group selfies or group photos. The phenomenon of uploading photos over social network is enhancing widely [2]. Besides, People are curious to upload their images over social network apps captured during an Event [2]. Therefore, after just an Event ends, they keep on asking mobile owner (one who captured the photos) to send an individual his photos.

This goes very hectic for mobile owner to identify photos of each individual and send them their photos separately. Mobile Owner has to find the photos of each individual in the Gallery and send them separately. Not only this, a third-party application like WhatsApp, Messenger or Bluetooth is required to send those photos, making extra burden for mobile owner. There are many applications (23Snaps, Cluster, Path, letmesee etc.) available that share photos with small circle of friends but they require user's interaction to identify individuals in the photo. However, SnapShare is the only application that uses Face Recognition Algorithm to classify individuals in the photos and automatically shares photos with recognized and require no user interaction of user for photo sharing.
SnapShare provides relief to both sender and receiver. SnapShare is Deep Learning and Machine Learning based Human Face Recognition System having Client-Server Architecture. SnapShare has an Android Application as its Client tier and Python based Server for Face Recognition and Storage. Client App communicates with Server using Android networking libraries (Google Volley [3] and Retrofit [4]). Client App captures and send photos to Server, Server classifies (Recognizes) each face present in the photo and share that photo with all classified (recognized) users.

The scope of project includes: capture and store users' photos, classify (recognize) individuals present in the photo, provide users access with their photo no matter which mobile phone photos were captured. The aim of the SnapShare is to ease mobile owner and people present in group photo to share and access photo automatically. For mobile owner, he does not need to categorize and share each individual's photos. For photo members, they no more have to ask mobile owner for their photos. The specified objectives of this application is to develop an android camera app that captures and upload the photos on server. Then classify (recognize) the individual in photos using face recognition algorithm. After classifying images application auto-share the photos with recognized individuals.

2. Methodology

This section illustrates different activities that are concerned with implementation of SnapShare. Iterative Development approach is used for development of SnapShare. In iterative development, feature code is designed, developed and tested in repeated cycles. With each iteration, additional features were designed, developed and tested until full functional SnapShare was ready for use.
For development of SnapShare following tools are used: android studio, PyCharm, Ubuntu server, apahe2 server, Microsoft azure and MySQL. Client application is developed using android studio, whereas client application is developed using python language. Two servers are used for server side services: Ubuntu server and apahce2.

SnapShare is a two-tier application i.e. it has Client-Server Architecture. Client App has following features: user sign up, user login, capture photos, uploading images to server, and downloading images to server. In order to train application it uses 8 to 10 pictures of user in first step (sign up). Two libraries are used for sending user data and images to server. Volley library [3] is used to send user name, email and Retrofit library is used to upload images on server. Retrofit is used here because retrofit send image in multipart without requiring to convert image to base64. For integrating camera into application fotoapparat [5] library is used. Fotoapparat gives us below listed features.

Using android Intent Service, all captured photos will be sent to server in retrofit [4] multipart for classification of human faces in the images in the background. User can view the images that he/she has uploaded or the images that he is present in. We have used Picasso android library [6] to download and cache images.

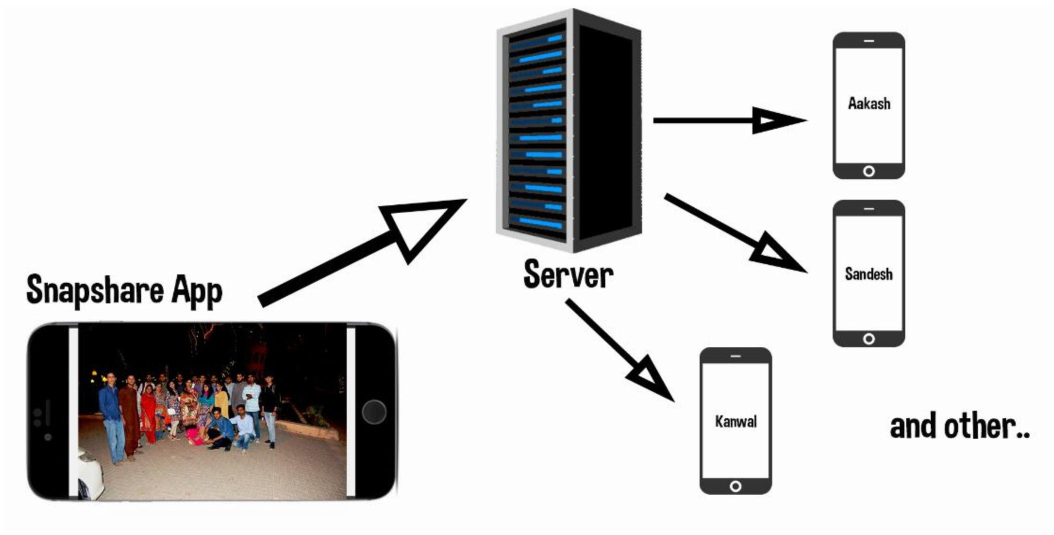

FIGURE 1. SnapShare client architecture

Server performs following computations on received image: face detection, identification of face landmarks for feature extraction, face transformation to align faces, predicting individual's name using trained classifier, and sharing of image.

SnapShare server detects all the faces present in the image using HOG (Histogram Oriented Gradient) algorithm. Implementation of HOG algorithm is present in DLIB library [7]–[9]. So, SnapShare server uses HOG algorithm from DLIB library. Face detection crops the detected faces from the image and passes it to Face Landmarks algorithm. Face landmarks algorithm detects the exact location of face features such as eyes, lips, nose, chin etc. Face Landmarks Estimation Algorithm yields 68 specific points (landmarks) that exist on every face. We have used Vahid Kazemi and Josephine Sullivan algorithm [10] for Face Landmarks Extraction, its implementation is available in Openface library. Faces in images can be in different orientations like side poses. So, in order to be able to recognize faces in different orientations we have to align those faces to make them ideal for deep convolutional neural network (CNN) [11], [12]. We are using affine transformation [13] for the alignment of faces because it does not cause distortion of image while transforming it to best possibly centered. Face alignment algorithms uses Face landmarks to align those faces in ideal position. After alignment, those cropped faces are suitable for deep CNN. SnapShare uses Openface's pre-trained deep convolutional neural network which is trained on LFW (Labeled Faces in the Wild) dataset which contains 500,000 images of faces. Openface's deep CNN outputs unique 128 measurements against each face, these measurements are also called Face Embedding. In short, these 128 measurements are the representation of face in numbers. This approach of using Deep CNN for face recognition is developed by researchers at Google in 2015 which is published as Google's FaceNet [14]. TensorFlow [15] library is used to operate on Deep CNN. Generated 128 face's measurements (Face Embedding) are stored in MySQL database with the name of the person. SnapShare server trains the Radial SVM classifier on those Face Embedding and name of the person as class label. Radial SVM's implementation is present in Scikit-learn library of Python [8], [16], [17], [18],[19], [20]. To predict the names of the individuals present in the image, SnapShare server repeats the above mentioned 1 to 4 steps and then passes those 128 measurements to the trained SVM which yields the name of the person against those Face embedding. After predicting the names of the all the individuals in the image, SnapShare server give access of that image to all the recognized individuals. User can access all his/her images in the Cloud gallery of SnapShare Client App.

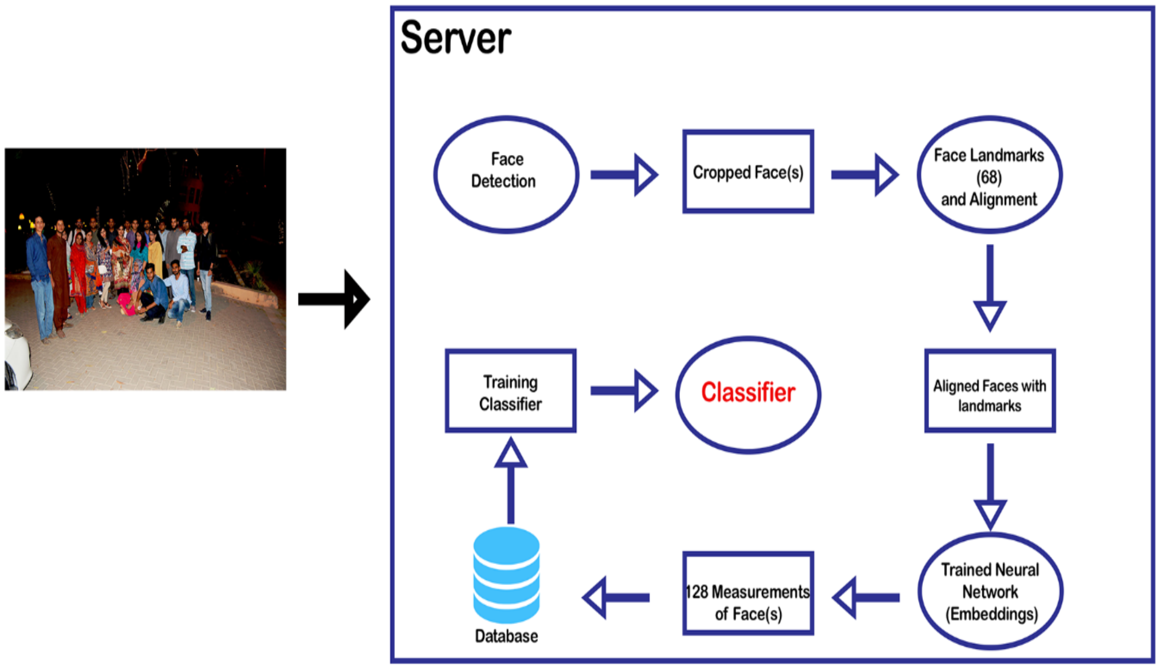
Server
Face Detection
Cropped Face(s)
Face Landmarks (68) and Alignment
Training Classifier
Classifier
Aligned Faces with landmarks
Database
128 Measurements of Face(s)
Trained Neural Network (Embeddings)

FIGURE 2. SnapShare Server Architecture

SnapShare server provides REST API to its client app which is used for communication and image transfer between the SnapShare client App and SnapShare server. SnapShare server is completely implemented using Python3/Flask framework and MySQL is used as database. SnapShare server is hosted on Apache2 under the WSGI module. Following diagram overall architecture and design framework of our system.

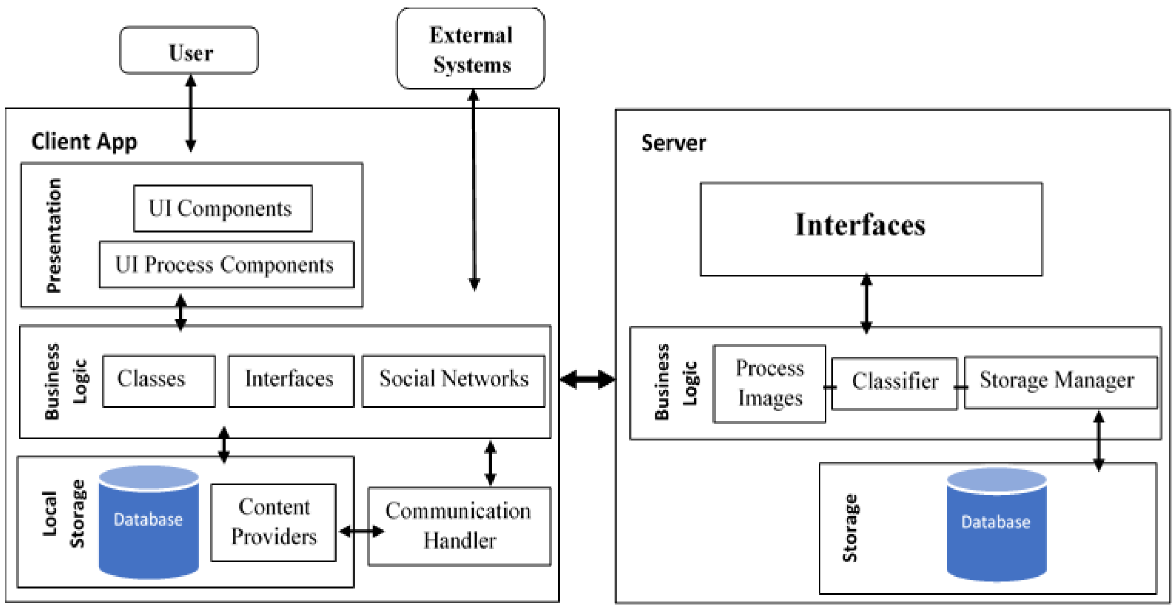

FIGURE 3. SnapShare Design Framework

3. Results and Discussion

This study shows the importance of picture sharing in these days. SnapShare system has been developed using client-server architecture as a solution of indicated gap. Ten people were selected in this study for testing of the system. People registered themselves in application and train the system with their images. Total number of images used for training the people were twenty. Pictures were taken on different location with different people. Three category of pictures were used to check the accuracy of system. First category consists of ten people in a group. Second category consists of five people in group. Last and third category consist of single people in an image. Following table describes the result of our system on those ten people.

PERSON	IMAGES FOR TRAINING	IMAGES TAKEN IN GROUP OF TEN (10)	CORRECT CLASSIFIED	IMAGES TAKEN IN GROUP OF FIVE (5)	CORRECT CLASSIFIED	SEPARATE IMAGE	CORRECT CLASSIFIED
PERSON 1	20	10	9	12	11	11	11
PERSON 2	20	12	11	10	10	10	10
PERSON 3	20	10	9	5	5	6	6
PERSON 4	20	12	12	7	7	8	7
PERSON 5	20	9	9	9	8	11	10
PERSON 6	20	14	12	12	10	12	11
PERSON 7	20	20	19	7	7	8	8
PERSON 8	20	10	9	8	7	7	7
PERSON 9	20	9	8	12	11	12	11
PERSON10	20	15	13	6	6	6	6

TABLE 1. Results of the System from different category of images

Above table shows the result on all categories of images. So, in first category of ten people minimum nine images were used to test the accuracy. All images of person 5 were correctly classified. Whereas, image of person 9 gets 90% accuracy approximately. Twenty images (maximum) from same category were used to test the accuracy. From them nineteen were classified correctly, it means system gives more than 90% accuracy. On other hand minimum six images were selected to test the accuracy from category of five people. All six images were classified correctly. Maximum twelve images were selected from same category. Twelve images of three persons were used. Eleven images of two person

(person 1 and person 9) were classified correctly. Ten images of person 6 were classified correctly. Last category of single image of people have more accuracy than others. From maximum twelve images, eleven images were classified correctly. Below figure describes same above data of table in graphical form.

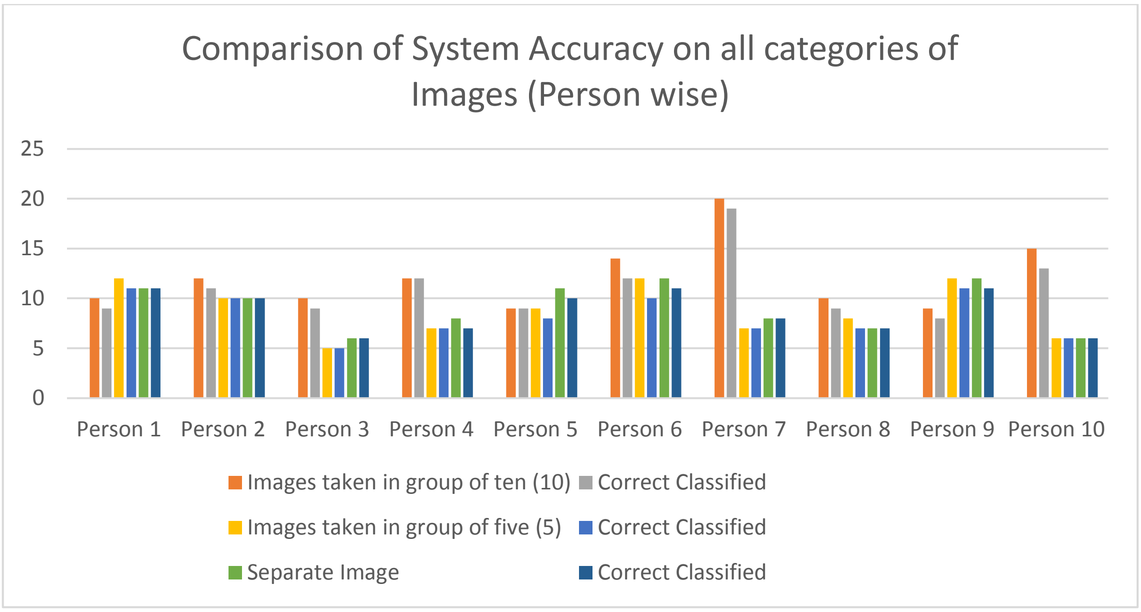

FIGURE 4. Comparison of System Accuracy on all categories of Images (Person-wise)

Finally, from above discussion, it is clear that category of an image having single people have more accuracy than other categories. One twenty one (121) images were used in first category of 10 people group. One hundred eleven (111) were correctly classified from them. Eighty eight (88) images were used in second category of 5 people. Eighty Two (82) were correctly classified.

At last from third category, Ninety one (91) images were used and Eighty Seven (87) were classified accurately. 91.7%, 93.2%, and 95.6% are accuracy of three categories group of 10 people, group of five people, and group of single person respectively. Below figure describes the accuracy of three categories.

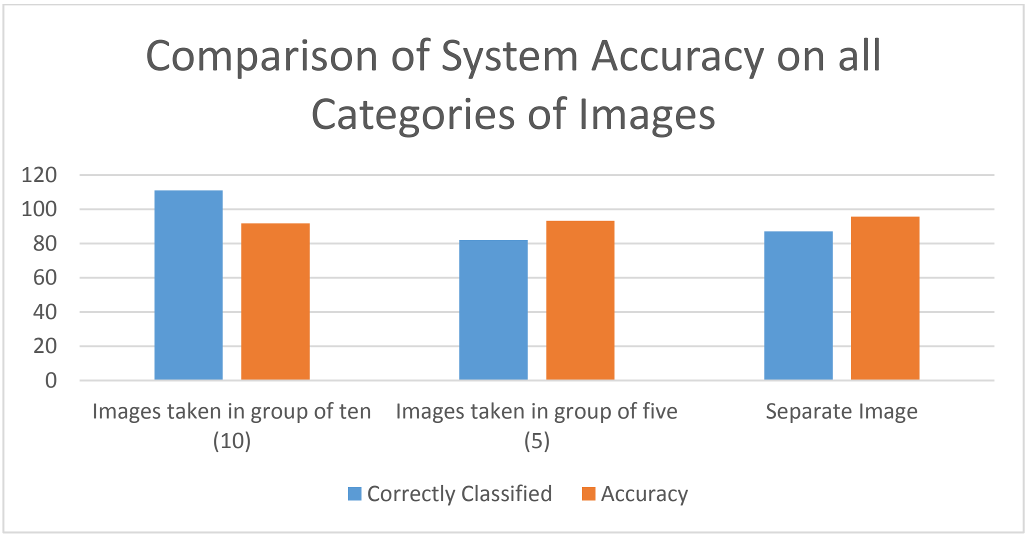

FIGURE 5. Comparison of System Accuracy on all categories of Images

4. Conclusion

There are rare events we miss without having photographs. Capturing photos with friends or family is fun but sending everyone their photos separately and using different medium goes hectic. A sender has to identify photos of each individual and distribute right photos to right persons. The solution we proposed for the described problem is called SnapShare. SnapShare relieves mobile owner from sending photos in a way that a receiver no more has to identify photos of his friends and them separately. But the only thing that user is required to do is capture photos with friends which is fun. SnapShare is Machine Learning and Deep Learning based System having Client Server Architecture. The client tier provides user interface to capture photos and server process image captured using client app and yields a name against each face present in the photos. In turn, all the classified users would get their photos automatically.

The work that requires to be done is future is:

- SnapShare iOS version is to be developed.
- We will be integrating social media in SnapShare. So that user could share his photos to social media using SnapShare directly.
- SnapShare works fine on Wi-Fi network. However, we need to make it work fine on 2G/3G/4G network as well.

References

[1] G. Rose, "Practising photography: an archive, a study, some photographs and a researcher," J. Hist. Geogr., vol. 26, no. 4, pp. 555–571, 2000.

[2] T. Correa, A. W. Hinsley, and H. G. De Zuniga, "Who interacts on the Web?: The intersection of users� personality and social media use," Comput. Human Behav., vol. 26, no. 2, pp. 247–253, 2010.

[3] Y. Shulin and H. Jieping, "Research and implementation of Web Services in Android network communication framework Volley," in 2014 11th International Conference on Service Systems and Service Management (ICSSSM), 2014, pp. 1–3.

[4] M. Lachgar, H. Benouda, and S. Elfirdoussi, "Android REST APIs: Volley vs Retrofit," in 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), 2018, pp. 1–6.

[5] "GitHub - RedApparat/Fotoapparat: Making Camera for Android more friendly. ??" .

[6] Y. SONG, S. OU, and J. LEE, "An Analysis of Existing Android Image Loading Libraries: Picasso, Glide, Fresco, AUIL and Volley," DEStech Trans. Eng. Technol. Res., no. imeia, 2016.

[7] D. E. King, "Dlib-ml: A machine learning toolkit," J. Mach. Learn. Res., vol. 10, no. Jul, pp. 1755–1758, 2009.

[8] A. R. Gilal, M. Omar, and K. I. Sharif, "A RULE-BASED APPROACH FOR DISCOVERING EFFECTIVE SOFTWARE TEAM COMPOSITION," JICT, pp. 1–20, 2014.

[9] A. R. Gilal, O. Mazni, J. Jafreezal, S. Kamal Imran, A. W. Mahessar, and B. Shuib, "Software Development Team Composition: Personality Types of Programmer and Complex Networks," in Proceedings of the 6th International Conference on Computing and Informatics, ICOCI 2017, 2017.

[10] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1867–1874.

[11] O. M. Parkhi, A. Vedaldi, A. Zisserman, and others, "Deep face recognition.," in bmvc, 2015, vol. 1, no. 3, p. 6.

[12] A. R. Gilal, J. Jaafar, L. F. Capretz, M. Omar, S. Basri, and I. A. Aziz, "Finding an effective classification technique to develop a software team composition model," J. Softw. Evol. Process, vol. 30, no. 1, 2018, doi: 10.1002/smr.1920.

[13] P. Dong and N. P. Galatsanos, "Affine transformation resistant watermarking based on image normalization," in Proceedings. International Conference on Image Processing, 2002, vol. 3, pp. 489–492.
[14] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 815–823.
[15] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," in 12th Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, pp. 265–283.
[16] B. Scholkopf et al., "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," IEEE Trans. Signal Process., vol. 45, no. 11, pp. 2758–2765, 1997.
[17] A. R. Gilal, J. Jaafar, M. Omar, S. Basri, and A. Waqas, "A rule-based model for software development team composition: Team leader role with personality types and gender classification," Inf. Softw. Technol., 2016, doi: 10.1016/j.infsof.2016.02.007.
[18] A. Abubakar et al., "A support vector machine classification of computational capabilities of 3D map on mobile device for navigation aid," Int. J. Interact. Mob. Technol., vol. 10, no. 3, 2016, doi: 10.3991/ijim.v10i3.5056.
[19] A. Waqas, A. R. Gilal, M. A. Rehman, Q. Uddin, N. Mahmood, and Z. M. Yusof, "C3F: Cross-Cloud Communication Framework for Resource Sharing amongst Cloud Networks: An Extended Study," Int. J. Comput. Sci. Netw. Secur., vol. 17, no. 8, pp. 216–228, 2017.
[20] A. Waqas, A. W. Mahessar, Z. Bhatti, N. Mahmood, M. Karbasi, and A. Shah, "Transaction Management Techniques and Practices in Current Cloud Computing Environments : A Survey," Int. J. Database Manag. Syst., 2015, doi: 10.5121/ijdms.2015.7104.