Face Recognition using Triplet loss function in Keras

Akhil Gorijavaram, Ramanathan L, Hemn Barzan Abdalla, Prabhakaran N, Ramani S, Rajkumar S

Abstract**:** ***Face recognition could be a personal identification system that uses personal characteristics of an individual to spot the person's identity. face recognition procedure primarily consists of 2 phases, firstly face detection, which identifies the portion of face in an image, second is the recognition, that recognize a face as people. In 2015 FaceNet introduced a new method [1] for face recognition achieving a new record accuracy at that time. The essence of the idea is to map the face images into a 128-dimensional embedding on a unit hypersphere. The relation between two pictures can be determined from the distance of their embeddings. If two embeddings are close to each other that means the persons on the pictures look similar. This was done in tensorflow, there are many algorithms[2] such as OpenFace[12] which tried to take FaceNet as the basis and tried to improve the results. Our goal is to create an implementation of the FaceNet solution in Keras, a deep learning library and to generate visualization for the 128th dimensional representation of the face images using the newly released UMAP algorithm[4].***

Index Terms**:** ***FaceNet, keras, triplet loss, UMAP***

I. INTRODUCTION

Biometric based authentication has been widely popular to identify a person, fingerprint is used in mobile phones, offices etc. The main reason is they are reliable, hard to forge unlike a normal hand-written signature. There are other ways to authenticate a person such as password, PIN, smart card etc. The issue with these is these can be stolen or cracked, once it is stolen it can be used for wrong purposes, it is better to use a biometric system than any other methods. They cannot be stolen or forged lives or dies with the person, there are many ways to use biometric authentication such as fingerprints, iris scan, voice, face recognition, hand. Fingerprint is the most popular, every phone now a days has a fingerprint scanner, but the issue is we drop our fingerprints everywhere, so it is easy to get a person's fingerprints from his things or other stuff. Iris scanner's equipment is costly and not that reliable, it can be really buggy. These reasons made face recognition a viable and better option. Face recognition has become popular because of its wide range of applications, Face Recognition is derived from object detection. it helped robots to identify the person, it can be used by surveillance cameras to identify any unauthorised intruders. It can be used in offices, colleges for attendance purposes. It is helpful in tracking a person. The main objective is for a computer to identify a person accurately and quickly in real time under various difficult situations. Face recognition can use both static photos and videos. Face recognition works by using features of the face which will remain the same after years, even with beard, spectacles, even if you gain or lose any weight. There are some complications because a photo of person can be influenced by many things such as light, angle, expressions, poses, photo size etc. There are many types of face recognition such as feature based, image-based. In feature based we use skin color, facial geometrics, face landmarks etc to identify a person. Image based type of facial recognition uses statistical and neural networks. Fa Face recognition could be used in various sectors like Military, Commercial, Authentication, Law enforcement, Banking, Smart technologies, Security [16]. Face recognition research could be divided into holistic, feature-based methods. Initially face recognition was done using feature-based which involves considering distances, ratios, areas and other geometrical calculations. These were not accurate and couldn't be reliable. Later holistic methods started to popup which uses deep learning, neural networks, artificial intelligence etc. these learn information from the images and use this information to identify the person's identity. Statistical methods like Principal Component Analysis (PCA) convert faces into eigenvectors. Eigenfaces and fisherfaces are methods in PCA-based face recognition. Lawrence et al. demonstrated an AI method which is powered by CNN for classifying a face [11]. The aim of this project is to test different algorithms for facial recognition along with our own algorithm. Each algorithm is going to be trained with the identic information set. Then, another data set will be used to test the accuracy of the algorithms. Finally, the test results will be compared, and the we get to know the accuracy of our algorithm compared to others

EXISTING ALGORITHMS

A. Face Detection Algorithms:

1. Viola and Jones

The name comes from the authors, designed by Viola & Jones was one of the first real-time object (and face) detection method. The algorithm is based on three main concepts: Haar features and the integral image, feature selection via Adaboost and the attentional cascade. To summarize, a sliding window is passed over the image on which face detection is performed and for each of the sliding window's position, features are generated from the pixels in the sliding window.

Manuscript published on 28 February 2019.
***** Correspondence Author (s)

Akhil Gorijavaram*, Computer science and engineering, VIT, Vellore, India.
Ramanathan.L, Computer science and engineering, VIT, Vellore, India.

Then these features are passed through a cascade of classifiers to identify if these pixels correspond to a face [17].

2. HOG Histogram of Oriented Gradients (HOG for short) technique was presented as an object, more specifically in this case, human detection method in the paper "Histograms of Oriented Gradients for Human Detection". The idea behind this rule is that "local object look and form will usually be described by the dispersal of native intensity gradients or edge directions, even without precise information of the corresponding gradient or edge positions." The goal of the algorithm is then to produce a vector of features for a given image sub window based on those gradients and to use those features to classify a sub window as human or not. The complete feature extraction procedures can simply be transposed to face detection, the only difference being that a new classifier must be trained with the features [14].

B. Face Recognition Algorithms:

1.Dlib

Dlib, which is an open-source library containing collection of machine learning algorithms. These algorithms are written in c++, which are also available in python solve the real-world problems. It is utilized in each industry and academics in a very wide selection of domains as well as artificial intelligence, embedded devices, mobile phones, and enormous large performance computing environments". During an update trying to introduce "deep metric learning tooling", a face recognition program was added to the library. However, Dlib's documentation is really scarce. Applied examples are the only source of information with some undocumented prototype [14].

2.OPENFACE:

OpenFace face recognition library is an effort to bridge the accuracy gap that exists between state-of-the-art private and publicly available face recognition systems. In this paper, they briefly present the application of deep learning to face recognition and explain the main ideas of two deep learning approaches called DeepFace and FaceNet. They finally describe their own method based on FaceNet. I will therefore first describe the FaceNet approach developed by researchers at Google and considered as a breakthrough paper by many public media and then go over the modifications brought in the context of OpenFace. FaceNet learns directly from face images using Euclidean space where distances depend on face similarity". The feature extraction is done using a deep neural network as in previous work such as DeepFace or DeepId2+. The authors specify that their method achieves greater representational efficiency" because the output of their method is directly a 128-D embedding" while previous methods used an intermediate bottleneck layer as representation leading to 1000s of features. To achieve this, the authors do not consider the problem as a mere classification problem and use a specific loss function that they call triplet loss [12].

3.ARCFACE

ArcFace is the name of a loss function and of the algorithm that use it as described in ArcFace: Additive Angular Margin Loss for Deep Face Recognition" which is one of the most recent research paper on face recognition (Jan 2018). As with each new paper on face recognition, the goal of the authors was to improve the state-of-the-art results. To do so, they identify three main attributes that differentiate face recognition models: the training data employed to train the model, the network architecture and settings and the design of the loss functions. They then try to work on these three attributes [10].

II. PROPOSED METHOD

This can be achieved by a Siamese network [6]. This idea comes from the DeepFace paper released at 2014 [2]. Their method reached an accuracy of 97.35% on the Labeled Faces in the Wild (LFW) dataset [9]. In 2015 FaceNet continued this gain by reaching a new record accuracy of 99.63% on the same dataset [1]. Their new approach was in the training process. They trained their network with the triplet loss function.

One of the most well-known open source implementations of the FaceNet is called OpenFace [12]. This was created using python and torch. With their latest release in 2016 their best model had an accuracy of 92.9%, improving their latest one from 76.1%.

When researching this topic, we found that there aren't many representations of the FaceNet in Keras [1]. The ones we have found were either in Tensorflow or their purpose were only to load in a pretrained model. So, we decided to create the whole training process in Keras.

A. *Triplet Loss*

The triplet loss function requires three images. Two images from the same person and a third one from a different person. The first picture is called the anchor image, the second is the positive image and the last is the negative.

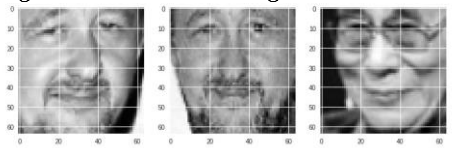

Fig.1. An example for a triplet

Their embeddings are 128th dimensional vectors that are constrained to live on the unit hypersphere. The task of the loss function is to make the distance between the anchor and the positive images small and the distance between the anchor and the negative big. Thus, the goal is to achieve the following inequality:

$$\left\|f(x_i^a) - f(x_i^p)\right\|^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|^2$$

Where the alpha is a margin between the positive and the negative distances. This alpha is necessary because otherwise the network would tend to put the embeddings at the same point causing the distances to become zero. The value that we want to minimalize is:

$$L = \sum_i^N [\left\|f(x_i^a) - f(x_i^p)\right\|^2 - \|f(x_i^a) - f(x_i^n)\|^2 + \alpha]$$

The problem is that if these triplets are generated randomly, after a short period of time the network will learn to distinguish them sufficiently.

Meaning the network can't learn from these triplets anymore. The solution to this problem is to teach it on triplets that are hard to distinguish [1].

B. Triplet Generator

There are three kinds of triplets:

- easy negatives: $d(a,p) + \alpha < d(a,n)$
- semi-hard negatives: $d(a,p) < d(a,n) < d(a,p) + \alpha$
- hard negatives: $d(a,n) < d(a,p)$

Easy negatives are the ones that are already satisfying the inequation. Thus, there would be no point to train on easy triplets because their loss value would be 0. We can train our network on semi-hard and on hard triplets. Although it is important to know that using hard triplets make the network more sensitive for bad data, for example mislabeled pictures. The generator's task is to generate this triplet. There are two ways this can be done:

- offline triplet mining: generating triplets every n step, using the most recent network weights on a subset of data
- online triplet mining: selecting the triplets from within a mini-batch. We used online triplet mining and mini-batches consisting of n people each with k pictures. Then we created all the possible combination for each person. And for all the possible pairs we chose one hard or semi-hard negative. Resulting in $k * (k - 1)/2 \cdot n$ triplet. If we couldn't find a right negative then that pair was discarded [1].

C. Network

For input size we chose 64x64 size grayscale images. The main motivation behind the smaller image size is to make the training process faster, and to allow our model to work well even on bad quality pictures. For the model we used a deep convolutional network [11] with inception modules [3]. It is based on the OpenFace's nn4.smallZ2.v1 network, which is a reduced version of the FaceNet nn4 model [1], [12]. We halved the filter numbers in the model significantly reducing its size. We also changed the pooling layers in the inception modules to match our input dimensions. And we had to remove all batch normalization layers because they caused the training to collapse. The training loss went down to 0 meanwhile the validation loss started to converge to the margin value meaning all the distances became zero. By adjusting its hyperparameters it didn't show any improvement so we decided to remove them. The model has three input and three output, for the triplet loss function. This architecture was done by the Keras functional API.

III. EXPERIMENTS

A. Steps for methodology

1. Dataset is first modified using align.py, removed any blurred or improper photos
2. OpenCV is used for face detection and we crop the dataset just to have faces
3. We give this data to our model, we can have four models, semihard triplets with 0.2 and 0.4 alpha, hard triplets with 0.2 and 0.4 alpha
4. We save the trained model, which is used on test data
5. Test data is also modified like train data and passed as input to the model
6. Finally, we calculate the accuracy and AUC values
7. We can use UMAP to visualize our trained model

B. *Datasets*

We used the VGGFace2 dataset as our training database [8]. The VGGFace2 is a large-scale face recognition dataset, which contains 3.31 million images of 9131 subjects. The subjects have a great variation in pose, illumination, age and ethnicity.We downloaded the "Train data_v1" and the "Test Data_v1" files from their website. We used a python program to process the data into a hdf5 file. The hdf5 extension was chosen because it provides a relatively fast way to access its content and it does not take up too much space on the hard disk. We decided that we will merge the train and test data, and later break it up into train and validation sets. For testing purposes, we chose another database, The Labeled Faces in the Wild (LFW) [9].
When we first created the dataset for the training, we used 40 images from each person. The face recognition was done with the OpenCV python module, based on the Haar Cascades. The "haarcascade_frontalface_default.xml" file was used as the cascade classifier. Those subjects, who were not recognized on their images at least 40 times were not used furthermore. The images were turned into grayscale images, and resized to 64x64 pixel size. Later we remade the dataset to improve the learning process. The images with text on them were filtered out, aligned in a way that placed the inner eyes and the bottom lip on the same place on every picture. We also filtered out the blurry pictures and those, which contained too much empty pixels. Our final dataset contains 7754 persons. The Labeled Faces in the Wild (LFW) dataset was used as our test data [9]. The LFW dataset contains 13233 images and 5749 people. The LFW data was pre-processed similarly to the training data, but if a picture could not be processed, we used another instance of the picture from the LFWcrop Face Dataset [13]. The images from this dataset was already cropped and resized to 64x64, so it was a good alternative.

C. Training

The training process was a very time-consuming process, so to get the best out of it, we made a lot of testing before training the final model. We tested several architectures. First, we had a small model with only four 3x3 convolutional layers and two max pooling layers after every second convolutional layer. Finally, it contained the fully connected layers to produce the 128 embedding. We used this model to comparison.
The next tests were to add inception modules to the network. Finally we decided to use Openface's nn4.small2.v1 network with some modifications, as we found that this network produced the best results [12]. The biggest modification was to halve the number of filters in every layer making it significantly smaller. This caused the training time to go down substantially with only a very little performance drop.

For optimizer we decided next to the Adam optimizer with the learning rate set to 0.0001. Our loss function was the triplet loss function. It doesn't have an implementation in Keras so we had to define it as a custom loss function. For the final training we had two hyper parameters that were still not tested:

- alpha: the margin in the triplet loss function
- negative triplet types: the triplet generation could be done in two ways, the generator could generate semihard or hard triplets

Using these hyperparameters, we could check all four combinations, and after the training, choose which one was the best model. The best validation loss was achieved by the model with 0.2 margin and semi-hard triplet generation.

TABLE I: Test Results

model	**Precision**	**Recall**	**Accuracy**	**AUC**
Semihard 02	0.8620	0.7972	0.8326	0.9085
Semihard 04	0.8198	0.8108	0.8127	0.8896
Hard 02	0.8533	0.8189	0.8365	0.9079
Hard 04	0.8301	0.7771	0.8064	0.8841

Since the project's goal was to create a good model for face recognition, the precision and accuracy values are both important. The "Semihard 02" model has the best precision rating, while its accuracy only marginally worse than the "Hard 02" model. Also, the AUC rating is also better for the "Semihard 02" model. So, we chose the "Semihard 02" as our best model.

D. Testing

The testing was made with the LFW dataset [9], [10]. There were 10 sets of matched pairs and mismatched pairs. We determined a threshold value for a set using 10fold cross validation. Using the determined threshold, we calculated the number of true positives, true negatives, false positives and false negatives based on our model's prediction. From this we gained an accuracy rating on the given set. The final accuracy rating was the average of the 10 set's accuracy ratings.

We also generated a ROC curve for each set, and an average ROC curve as well. We also calculated the AUC values, which provided us another feedback.

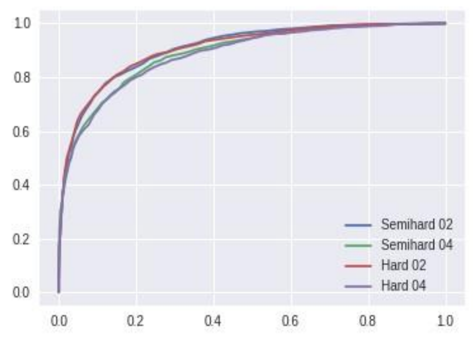

Fig. 2. The ROC curves of the models

E. Visualization

To visualize the training process, we used the newly released UMAP algorithm [7]. At the current time this is the most powerful dimension reduction tool that we know of. As the output of the model is a 128th dimensional vector it makes it suitable for this type of representation. We compress it to the 2D to make it easy to examine. For this purpose, we created a dataset from the LFW dataset containing 20 pictures from 20 people [9], [10]. If their embeddings are close to each other than the training were successful as the model sufficiently recognized the similarities in the pictures. First we created a diagram before the training to have comparison.

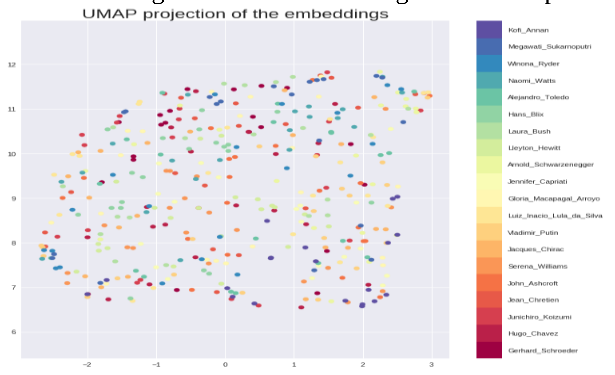

Fig. 3. The dimension reduction before the training

Finally, here is how the dimension reduction looks like with our best model.

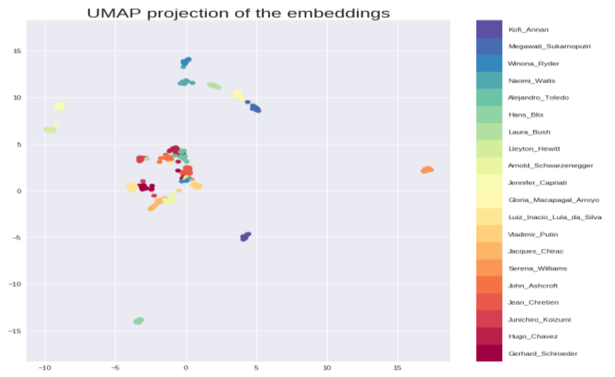

Fig. 4 The dimension reduction after the training

To make the figures more interactive we used the bokeh library to create plots with mouseover tooltips of the images.

IV. CONCLUSION

Face Recogntion is one of the most used biometric recognition system, Triplet loss function which was used in FaceNet gave remarkable results, so we tried the same in Keras environment. Normally we use trained models, but here we are creating our own model. The objective of this paper was to create a face recognition algorithm with Keras API and in this process we chose triplet loss function and adam optimizer for best results. In triplet loss function we used both hard and semi-hard triplets with margins 0.2 and 0.4 and we got best results with semi-hard triplets with 0.2 margin. This is just the start point, if we build the inception models in a different way, we might see interesting results, we could also change margin value and observe the results, the input image size we used was 64*64, increasing this might give us better results. overall, we got decent results, this is just a start point, maybe in future this can be improved to compete with other top algorithms, show that keras is a viable option.

REFERENCES

1. Florian Schroff, Dmitry Kalenichenko and James Philbin,
2. "FaceNet: A Unified Embedding for Face Recognition and Clustering" arXiv:1503.03832v3, 2015.
3. Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. "Deepface: Closing the gap to human-level performance in face verification." In IEEE Conf. on CVPR, 2014.
4. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions." CoRR, abs/1409.4842, 2014.
5. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision." arXiv preprint arXiv:1512.00567, 2015.
6. C. Szegedy, V. Vanhoucke, S. Ioffe and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning" arXiv:1602.07261v2, 2016.
7. G. Koch, R. Zemel and R. Salakhutdinov, "Siamese Neural
8. Networks for One-shot Image Recognition", 2015.
9. Leland McInnes, "umap Documentation", 2018.
10. Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi and Andrew Zisserman," VGGFace2: A dataset for recognising faces across pose and age" arXiv:1710.08092v2, 2018.
11. Gary B. Huang, Manu Ramesh, Tamara Berg and Erik Learned-Miller," Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments" University of Massachusetts, Amherst, Technical Report 07-49, October, 2007.
12. Jiankang Deng, Jia Guo, Niannan Xue, Stefanos Zafeiriou, ArcFace: Additive Angular Margin Loss for Deep Face Recognition, January 2018.
13. Guosheng Hu, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z. Li and Timothy Hospedales," When Face Recognition Meets with Deep Learning: an Evaluation of Convolutional Neural Networks for Face Recognition" arXiv:1504.02351, April, 2015.
14. Amos, Brandon and Bartosz Ludwiczuk and Satyanarayanan, Mahadev, "OpenFace: A general-purpose face recognition library with mobile applications" CMU-CS-16-118, CMU School of Computer Science, 2016.
15. Deep Neural Networks with Relativity Learning for facial expression recognition, Yanan Guo , Dapeng Tao, Jun Yu ,Hao XIOng, aotang, ac eng ao, 2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW).
16. Development of Deep Learning-based Facial Expression Recognition System, Heechul Jung, Sihaeng Lee, Sunjeong Park, IEEE 2015 south korea conference.
17. Facial Expression Recognition via Deep Learning, Abir Fathallah, Lotfi Abdi, IEEE conference 2017
18. Facial detection using deep learning, Manik Sharma, 2017, IOP.
19. Real-time personalized facial expression recognition system based on deep learning, Injae Lee, Heechul Jung, 2016 IEEE International Conference on Consumer Electronics (ICCE).

AUTHORS PROFILE

Akhil Gorijavaram did his b.tech in sree vidyanikethan engineering college in electrical and electronic engineering. He is currently doing his masters in computer science information security in vit vellore india. He is from Tirupati, Andhra Pradesh, India He is currently doing his internship in signify which was formerly known as Philips lighting. His current research interest are security, sensors, lighting devices.

Dr.Ramanathan L has received his B.E. in Computer Science & Engineering from Bharathidasan University, Tiruchirappalli, India, and M.E in Computer Science from Satyabhama, Chennai, India, the Ph.D. degree in Computer Science and Engineering from VIT University, Vellore, India. He is currently an Assistant Professor (Selection Grade) in VIT, Vellore, India. His area of interest is Data Mining, Internet of Things, Image processing, Database systems, Software Engineering, Cloud Computing, and Virtualization. He is having 14+ years of teaching experiences. He has published more papers in International Journals and Conferences. He is an Editorial board member/reviewer of International/ National Journals and Conferences. His ongoing research is on Prediction, Classification, and Clustering for Educational systems. He is a member of IACSIT, CSI, ACM, IACSIT, IEEE (WIE), and ACEEE

Hemn Barzan Abdalla holds a Ph.D. degree in the field of Communication and Information Engineering, He possesses one decade of experience in teaching and worked as a project assistant in various higher education places and also in Neusoft Institute, Guangdong with member in Institute of training and development in Sulaimani (KRG). He has published more papers in International level Journals and Conferences. He is an Editorial board member/reviewer of International/ National Journals and Conferences His main focus is on Prediction, Classification and Clustering for Educational systems. He has more than 100 project system for several places. His research interests include big data and data security, NoSQL, application.

Prabakaran is an Assistant Professor(Senior) at School of Computer science and Engg, VIT University, Vellore, India. He received his B.E and M.E degree in Computer science and Engg from anna university, chennai 2009 and 2011 respectively. He received his PhD from VIT, vellore in Computer science by 2017. His research activities are carried out in pervasive computing and context aware computing using sensors and networks

Dr. Ramani S has received his B.E. in Computer Science & Engineering from Madras University, Chennai, India, and M.E in Computer Science from Bharathidasan University, Tiruchirapalli, India and a Ph.D. degree in Computer Science and Engineering from VIT University, Vellore, India. He is currently working as an Assistant Professor (Senior) in VIT, Vellore, India. His area of interest is Data Mining, Database systems, Nature Inspired Algorithm, Predictive Data Analytics. He is having 10 years of teaching experience and 1 year as Build Engineer in IT industry. He has published more papers in International Journals and Conferences. His on-going research is on Prediction, Classification in healthcare applications and pattern mining. He is a member of CSI, IEEE.

Rajkumar Soundrapandiyan is presently working as associate prof at school of engineering and Engineering, Vellore Institute of Technology (VIT), Vellore, Tamil Nadu, India. He received BE in engineering and Engineering from anna University, Chennai, India in 2008, ME in computer science and Engineering from anna University, Chennai, Republic of India in 2010. He has completed ph.D. at Vellore Institute of Technology (VIT), Vellore, India in 2017. His research interest includes computer vision, visual perception, object detection, medical image process, infrared image process, biometrics, info concealment, network security. He has published over twenty research papers in acknowledged international conferences and journals. he's a Reviewer for several acknowledged journals like Computers in Biology and medicine, Computers, IET computer Vision, etc. he's a member of IEEE and life member of CSI.