



Face Recognition using Deep Convolution Neural Networks Based on Transfer Learning

1P. Sankara Rao

Assistant Professor, Department of Electronics and Communication Engineering, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, 531162, India
sankararao.palla@raghuenggcollege.in

2K. Amarnadh

Student, Department of Electronics and Communication Engineering, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, 531162, India
18981A0482@raghuenggcollege.in

3J. Anu

Student, Department of Electronics and Communication Engineering, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, 531162, India
18981A0453@raghuenggcollege.in

4K.V.S.Sanjay

Student, Department of Electronics and Communication Engineering, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, 531162, India
18981A0458@raghuenggcollege.in

5M. Pavan Kalyan

Student, Department of Electronics and Communication Engineering, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, 531162, India
18981A0495@raghuenggcollege.in

6J. Sunny

Student, Department of Electronics and Communication Engineering, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, 531162, India
18981A0455@raghuenggcollege.in

5985

Abstract

Face recognition is a one of the prominent biometric technique for the identifying or verifying individual. Face recognition technology has a wide range of applications, including authentication, access control, finance, criminal investigation, military, and daily life. In recent years, researchers have implemented various deep learning based approaches for face recognition. Deep learning, particularly deep convolutional neural networks (CNN), has gained prominence in face recognition, with several deep learning models. Transfer learning is a popular deep learning technique that uses a pre-trained neural network model rather than recreating the network from scratch on a new problem. Transfer learning approaches are more accurate, faster, and easier to deploy. In this paper,



we applied transfer learning based GoogleNet, and SqueezeNet deep learning architectures for face recognition. We developed our own dataset to train and test the proposed work. Finally, we got an overall accuracy of 89 % on GoogleNet and 93 % on SqueezeNet. The results show that our face recognition model outperforms the most advanced models.

Keywords: Face recognition, deep learning, Transfer learning, GoogleNet and SqueezeNet

DOI Number: 10.14704/nq.2022.20.10.NQ55598

NeuroQuantology 2022; 20(10): 5985-6000

1. Introduction

The field of machine learning has made major development in recent years. A significant improvement in the machine learning is "deep learning," which exploits deep networked architectures comprised of several linear or nonlinear transformations to model high-level data abstractions. Deep learning, also known as deep structured learning or hierarchical learning, is a subset of machine learning techniques that emphasizes the interpretation of data representation. Deep neural networks (DNNs) have made substantial advancements in various computer vision applications, including face recognition without constraints [1]. However, modern highly-accurate face recognition systems are often built on extremely CNNs [2-4], and as a result, they are composed of a succession of convolutional layers. Deep learning models require a substantial amount of computational resources, such as massive memory and powerful GPUs, in order to produce high-performance results. To circumvent these limitations, current research focuses on designing neural networks that are both small and economical without losing performance.

In recent years, the construction of lightweight deep neural networks has become popular for improving the speed-accuracy tradeoff [5-8]. GoogleNet [9], AlexNet [10], SqueezeNet [5], ShuffleNets [6, 7], and MobileNets [8,18], VGG models [20] are some of the most well-known neural network architectures for conventional image identification tasks, and they achieve

exceptional results. However, only a very few authors have offered precise lightweight facial recognition architectures. In this paper, we present GoogleNet and SqueezeNet, two novel lightweight architectures for face recognition. The results reveal that the SqueezeNet deep CNN model is more effective than the GoogleNet deep CNN method.

The remainder of this work consists of the following: The 2nd section describes the materials and methods utilized for this study. The 3rd section examines over- and under-fitting of training data. The 4th section describes the methodology and implementation procedure. Section 5 provides performance measures, while Section 6 discusses conclusions and future work.

2. Materials and methods

2.1. Convolution Neural Networks

In deep learning, a CNN is a specialized neural network designed to handle data via many array layers. CNNs are particularly suited for applications involving image recognition and are widely used for face recognition. Convolutional layers are the fundamental components that allow CNN to perform its wonders. In a typical image recognition application, a convolutional layer consists of a large number of filters for recognizing the image features. Figure 1 shows the various layers of CNNs, including the convolutional layer, the pooling layer, and the fully connected layer.

5986



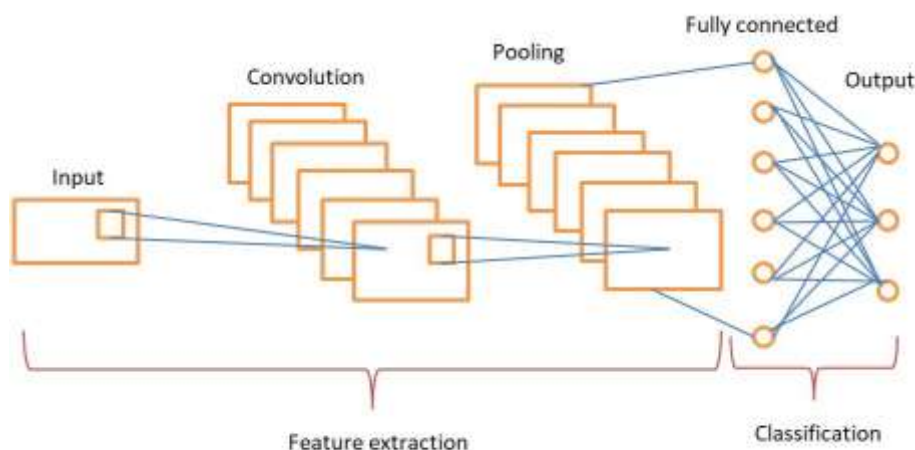


Fig. 1. CNN Schematic [17]

2.1.1 Convolutional layer

The essential component of a CNN is a convolutional layer. It includes a collection of filters (or kernels) whose parameters must be learned during training. Typically, the size of the filter is smaller than the image's size. In order to produce an activation map, each

filter is applied to the image. For convolution, the filter is slid across the image's height and width, and the dot product between each element of the filter and the input is computed at each spatial position. Figure 2 depicts the convolution process.

5987

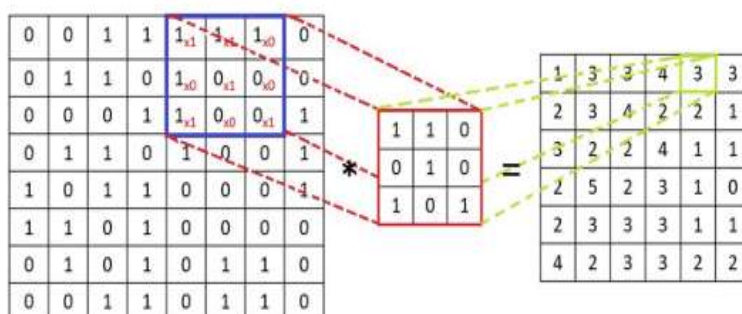


Fig. 2. An illustration of a computational activation map.

2.1.2 Pooling layer

Pooling layer is an essential component of CNN. CNN's pooling layer is used to reduce the size of the feature maps. It is advisable to reduce the set's number of values. The idea is that they will only extract relevant information and remove all irrelevant information. There are several ways to pool resources. Average pooling and Maxpooling

are the two most common pooling algorithms [11].

Average pooling: Average pooling computes the average of the components present in a region in a feature map. The area of the feature maps is governed by the size of the filter. The goal of average pooling is to calculate the mean of the patch's features. The average pooling process is depicted in Figure 3.



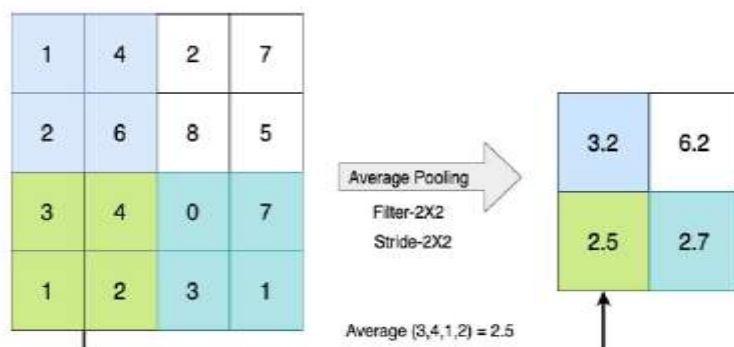


Fig. 3. An example of average pooling process

Max pooling: Max pooling is one of CNN's most used pooling algorithms [12]. During max-pooling, the maximum element from the feature map's region is chosen. As a result,

max-pooling produces the most prominent features from the preceding feature map. Figure 4 depicts an example of max-pooling.

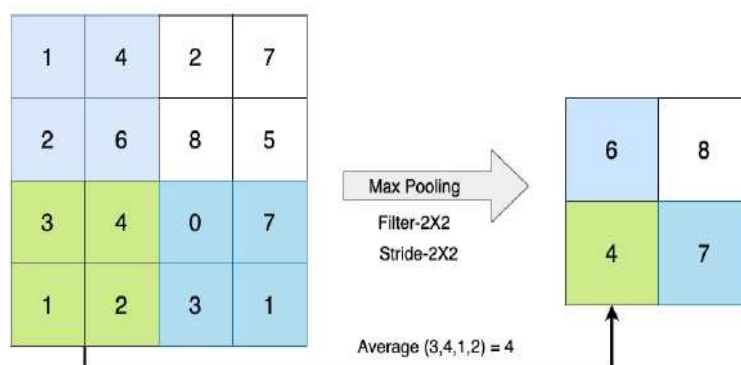


Fig.4. An example of Max pooling process

2.1.3 Fully Connected (FC) Layer

The FC layer is a CNN block found in the final network stages. It connects neurons from one layer to another. The FC layer receives the output of the pooling layer. Moreover, the FC layer is responsible for classifying the data into multiple categories.

2.2 CNN Pre-Trained Models

2.2.1 Pre-Trained GoogleNet CNN Model

GoogleNet is a 22-layer CNN initially presented at ILSVRC2014 (ImageNet Challenge) [13] hence its architecture is known as inception architecture. There are nine linearly stacked inception modules. The inception modules' terminals are linked to the global average pooling layer. It uses various techniques, such as 1x1 convolution and

global average pooling, to produce a more complex architecture

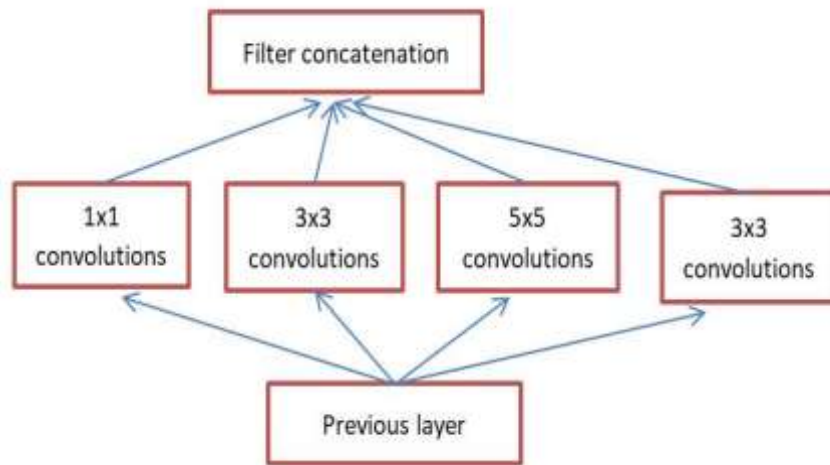
1x1 convolutions: The Inception architecture employs 1x1 convolutions in its design. These convolutions were used to reduce the amount of architecture parameters (weights and biases). By decreasing the parameters, we increase the architecture's depth.

Global Average Pooling: Global Average Pooling replaces the fully connected CNN layers. These GAP layers reduce the amount of model parameters to reduce Overfitting.

Inception Module: In this architecture, the convolution size of each layer is fixed. At the input of the Inception module, 1x1, 3x3, 5x5, and 3x3 max pooling are carried out in parallel, and their outputs are stacked to



produce the final result. Figure 5 depicts the naïve version of the inception module.



5989

Fig. 5. A naive version of an inception block [19]

Figure 6 demonstrates that the dimensionality reduction of the 11 convolution kernel can decrease the number of parameters and boost the network's depth.

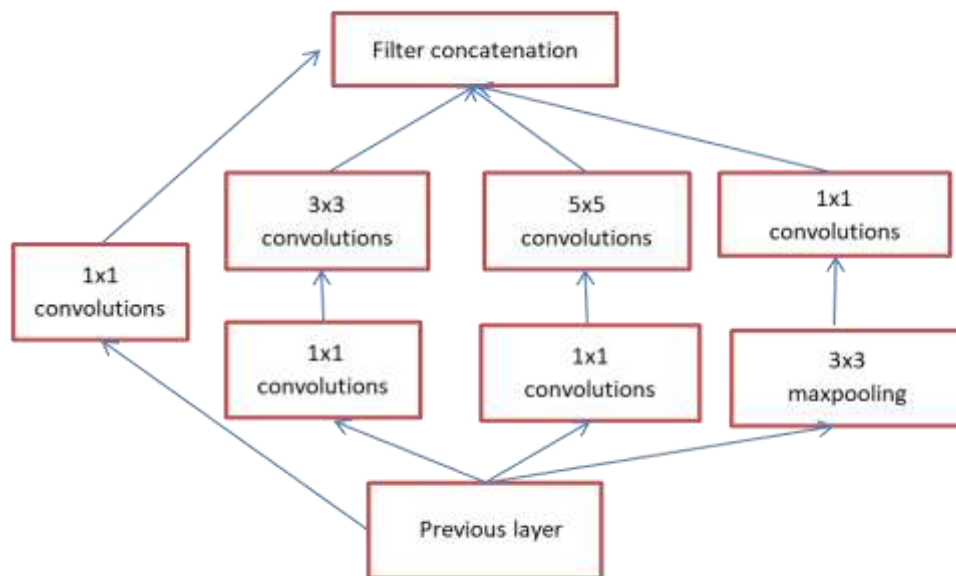


Fig. 6. Dimension reductions in the Inception module [19].

Figure 7 shows the comprised form of GoogleNet CNN.

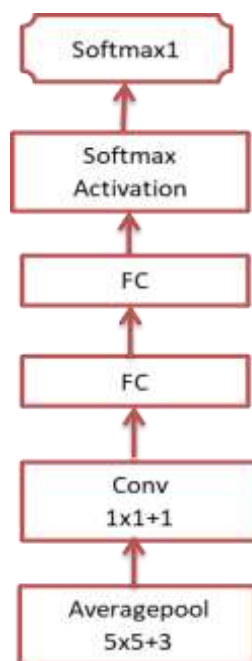


Fig. 7. Architecture of GoogleNet

2.2.2 Pre-Trained SqueezeNet Deep CNN Model

5990

SqueezeNet is a CNN architecture that reduces the number of parameters through the use of multiple design strategies. In comparison to AlexNet, we are able to reduce the size of our models by a factor of 50 using SqueezeNet, while still maintaining or improving upon AlexNet's top-1 and top-5 accuracy [5].

In this design, fire modules serve as the fundamental element. SqueezeNet is designed with three primary strategies: (1) Reduced network size by substituting 3x3 filters with 1x1 filters; (2) decrease in the number of inputs for the remaining three 3x3 filters; and (3) Late network down sampling so that convolution layers have substantial activation maps.

Figure 8 shows the fire module of the squeezeNet. The Fire module is made up of two layers: a Squeeze layer that reduces the number of input channels by using a limited number of 1x1 convolutions and an Expand layer that increases the number of output channels by using 1x1 and 3x3 convolutions. This technique is known as a bottleneck structure. In addition, the Fire module's Expand layer contains 1x1 convolution filters to further minimize the number of parameters. Figure 9 shows the architecture of the SqueezeNet CNN.



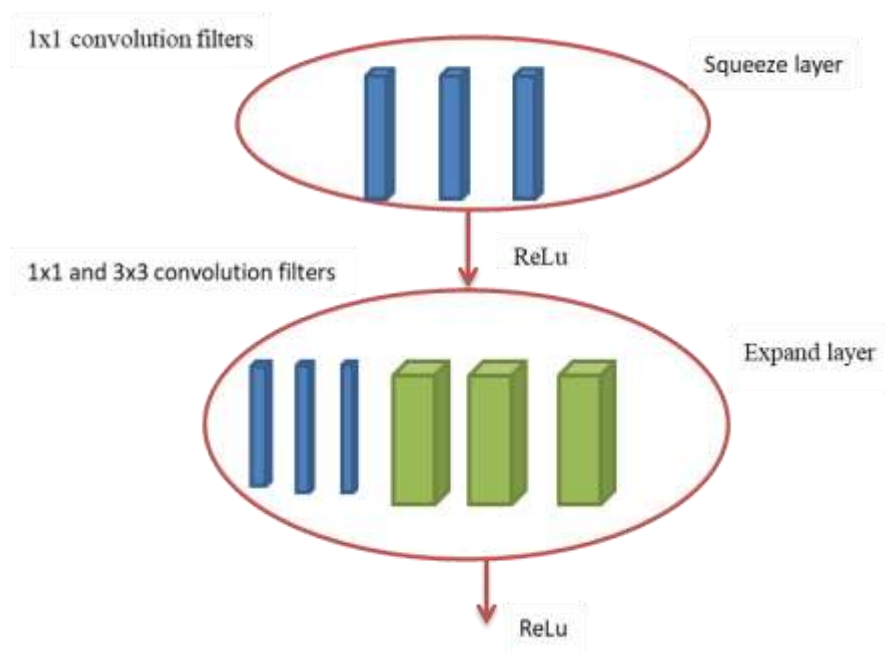


Fig. 8. SqueezeNet CNN Fire module

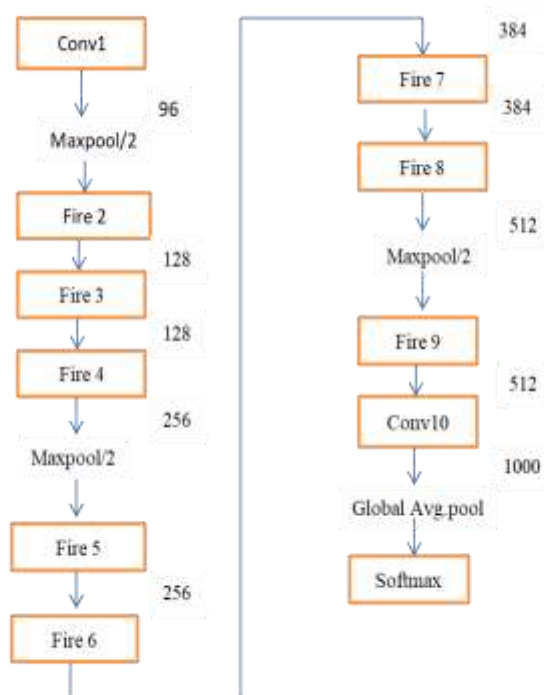


Fig. 9. Architecture of SqueezeNet CNN

3. Overfitting and Underfitting of training data

Deep neural networks have several parameters. With so many parameters, the network may fit a wide variety of complex datasets. This flexibility can cause Overfitting [14]. When a model is overtrained on a particular set of training data, it is more susceptible to overfitting. This type of model will perform exceptionally well on the training dataset, but its performance will not improve on the new or unseen dataset. This causes Overfitting. If the model performs poorly on both the training dataset and the new or unknown data, this is generally the result of underfitting.



Underfitting typically occurs when there is inadequate training time or when the learning algorithms are insufficient to model the training data. Figure 10 shows the underfitting, overfitting and optimal training data.

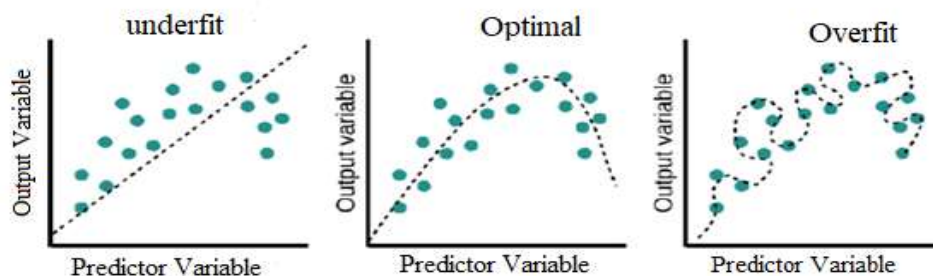


Fig. 10. Underfitting, optimal, and Overfitting of training data

Regularization is a technique for avoiding Overfitting and underfitting. Regularization is a machine learning approach, particularly employed in deep learning that adjusts the learning algorithm to improve model generalization to previously unseen data [15].

3.1 Early Stopping (ES)

Early Stopping is a regularization approach in machine learning that monitors the model's generalization error and helps in stopping the training process if the model's performance on the validation dataset begins to worsen. This prevents the training data from becoming overfit. This technique is the most often used kind of regularization in deep learning [16] due to its simplicity, effectiveness, and ease of implementation.

4. Methodology and implementation

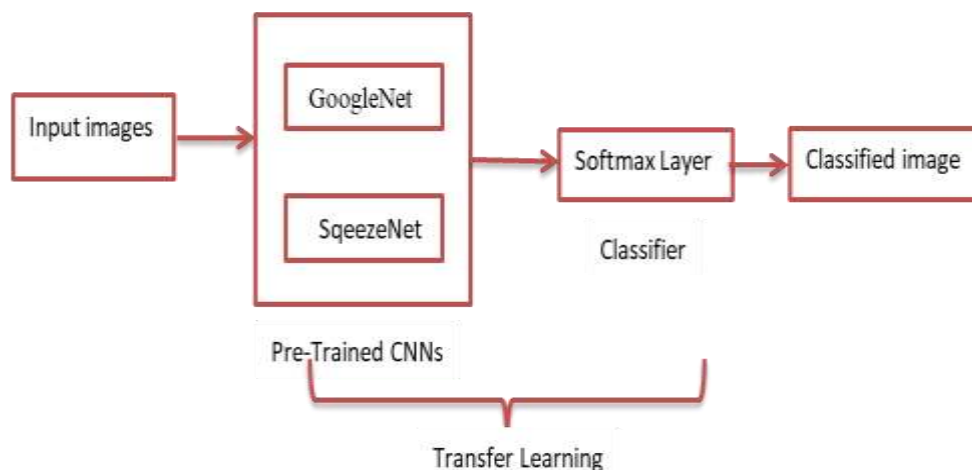


Fig. 11. Framework of face recognition system using pre-trained CNN transfer learning approach

The process begins with the acquisition of facial images from individuals, followed by the pre-processing of those images to achieve standard normalization. After that, the images are convolved using a pre-trained CNN transfer learning approach for feature extraction and classification, as shown in

Figure 11. In this paper, we applied GoogleNet and SqueezeNet pre-trained CNNs for face recognition.

4.1 Preparing the Dataset



The first step is to gather information. We looked at 40 photos of the 4 subjects to come up with this conclusion. For each subject in the database, the photographs are stored in a separate folder. There are four folders established here, each containing ten photos. All photos that may be included in the dataset are of varying dimensions. Consequently, pre-processing/image augmentation is necessary to ensure that the network accepts the training images as input images and prevents them from being over- or under-fit

4.2 Load the Dataset and pre-trained CNN

After data preparation, the next step is to load the data. The dataset is separated into

the training set and the validation set. We can use the 'splitEachLabel()' function in MATLAB to accomplish this. This function has two required arguments. The first argument represents the dataset to be split. And the ratio of splitting is the second argument. The ratio is set to 7/10. It indicates that the data will be divided into two groups, with one group having 70% of the data and the other obtaining 30% of the data. Based on the transfer learning model, the system loads the pre-trained network (GoogleNet/SqueezeNet), and the trained network takes these training parameters into account. Table 1 shows the various training parameters and its values.

5993

Parameters/Networks	GoogleNet CNN	SqueezeNet CNN
Mini batch size	10	4
Max epochs	6	10
Initial learning rate	0.003	0.004
Max.Iterations	42	70
Iterations for epoch	7	7

Table 1. Training parameters

4.3 Testing the Face Recognition Network

The network has undergone extensive training. Now we send images to testing network, which will classify them.

5. Evaluation Metrics

Model evaluation helps comprehend the model's performance. There are a number of evaluation measures available. Among these,

the confusion matrix is considered as an effective classification technique.

5.1 Confusion matrix

The confusion matrix demonstrates an algorithm's performance in terms of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) [9].

	PREDICTED VALUES	
ACTUAL VALUES	TP	FN
	FP	TN

Table 2 Confusion matrix

As seen in the Table 2, TP indicates that the model predicts exactly the presence of the positive class; TN denotes that the model accurately predicts the negative class; FP shows that model predicts the positive class incorrectly and FN specifies that model

predicts the negative class incorrectly. To evaluate the performance of GoogleNet and SqueezeNet CNNs in this paper, various metrics including precision, recall, F1 score, and accuracy are employed.



Precision: Precision is a metric that describes the accuracy of a model. It computes the overall number of accurate forecasts.

$$\text{Precision (P)} = \frac{TP}{TP+FP} \quad (1)$$

Recall: Recall is a metric that computes the total number of actual positives captured by a model.

$$\text{Recall (R)} = \frac{TP}{TP+FN} \quad (2)$$

F1-Score: The F1 score is a weighted sum of recall and precision.

$$\text{F1 - Score} = \frac{2 * P * R}{P + R} \quad (3)$$

Accuracy: Accuracy is the ratio of the number of correct predictions to the total number of data samples.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

6. Experimental results

The GoogleNet and SqueezeNet CNNs are trained and evaluated in the MATLAB R2021b environment.

6.1 Experimental results of transfer learning based pre-trained GoogleNet CNN

GoogleNet is one of CNN's pre-trained models. Figure 12 shows the pre-trained GoogleNet network architecture and specifications. Figure 13 shows face recognition results with 88.24 % validation accuracy. The operation takes 2 min 51 seconds to finish with a maximum of 42 iterations and 7 epochs/iteration. The constant learning rate schedule is 0.0003. The testing process achieves 86 % accuracy. Figure 14 depicts the testing result of GoogleNet.

5994

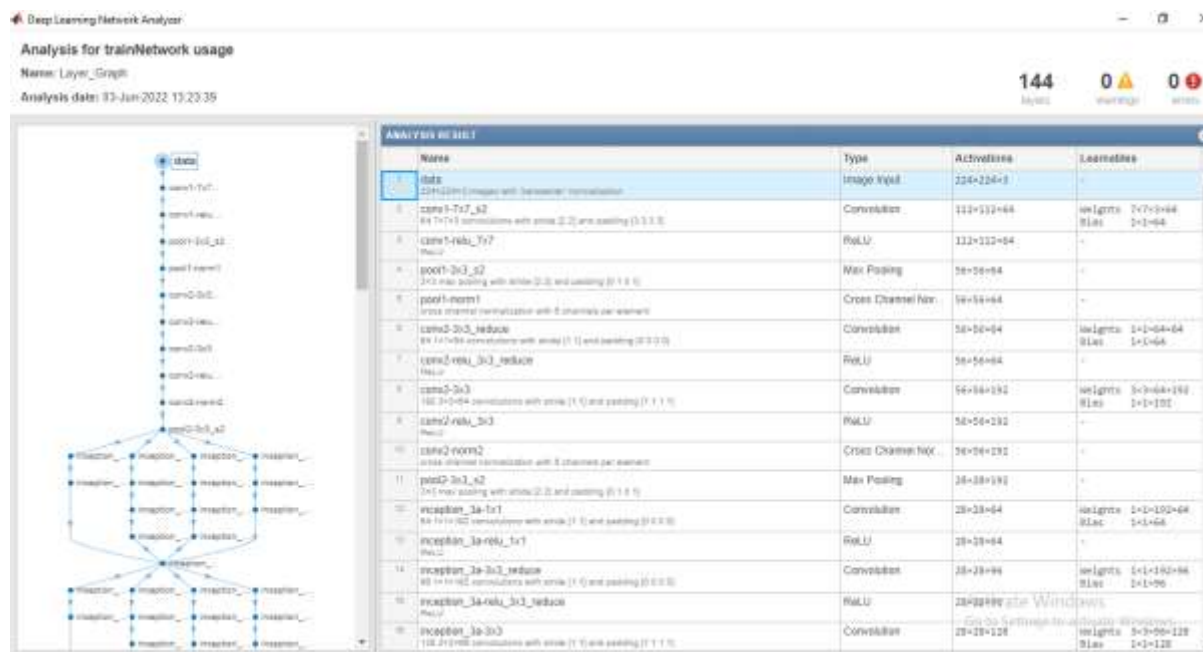
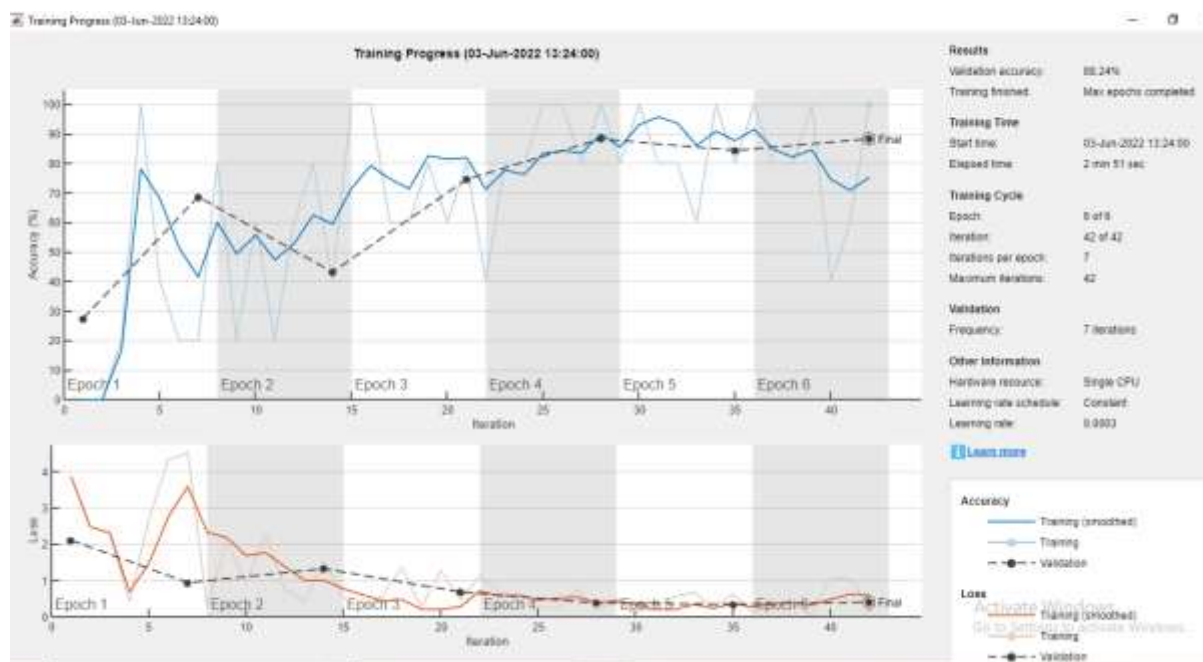


Fig. 12. GoogleNet CNN specifications and architecture (Figure continues)





5995

Fig.13. GoogleNet CNN's training progresses

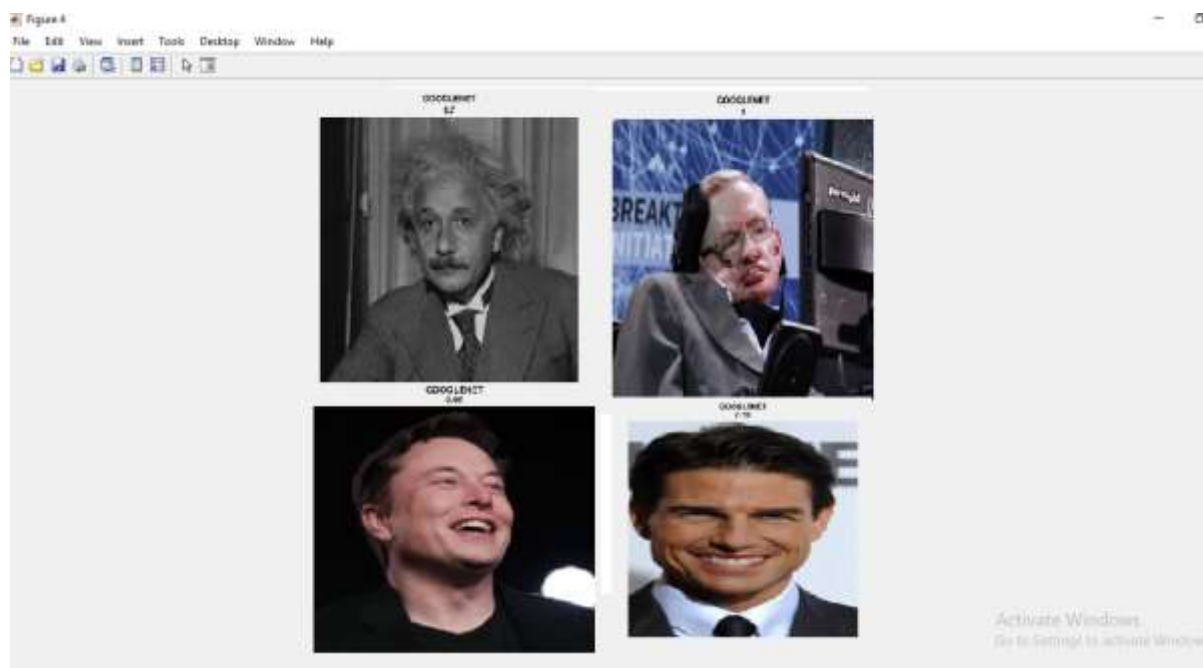


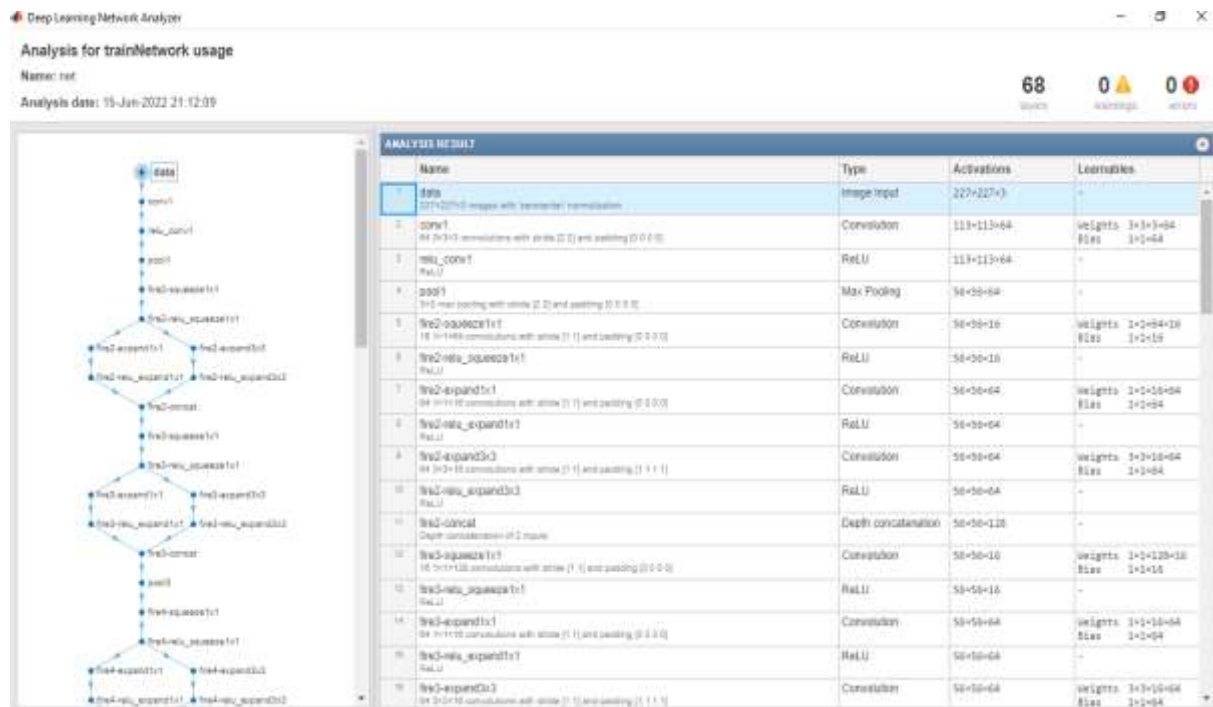
Fig. 14 GoogleNet CNN test progresses

6.2 Experimental results of transfer learning based pre-trained SqueezeNet CNN

SqueezeNet is another CNN's pre-trained models. Figure 15 shows the pre-trained SqueezeNet network architecture. Figure 16 shows face recognition results with 100 % validation accuracy. The operation takes 1min 51 seconds to finish with a maximum of 70 iterations and 7 epochs. The



constant learning rate schedule is 0.0004. The testing process achieves 93% accuracy. Figure 17 depicts the testing result of SqueezeNet CNN.
 List may be presented with each item marked by bullets and numbers.



5996

Fig. 15. SqueezeNet CNN specifications and architecture (Figure continues)

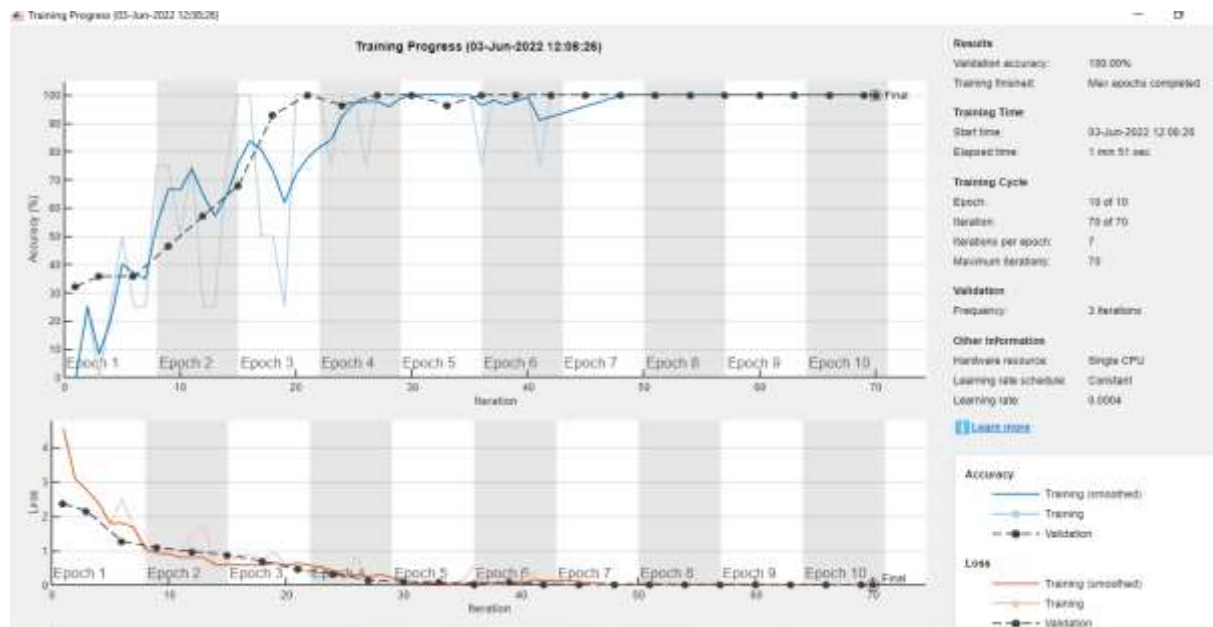
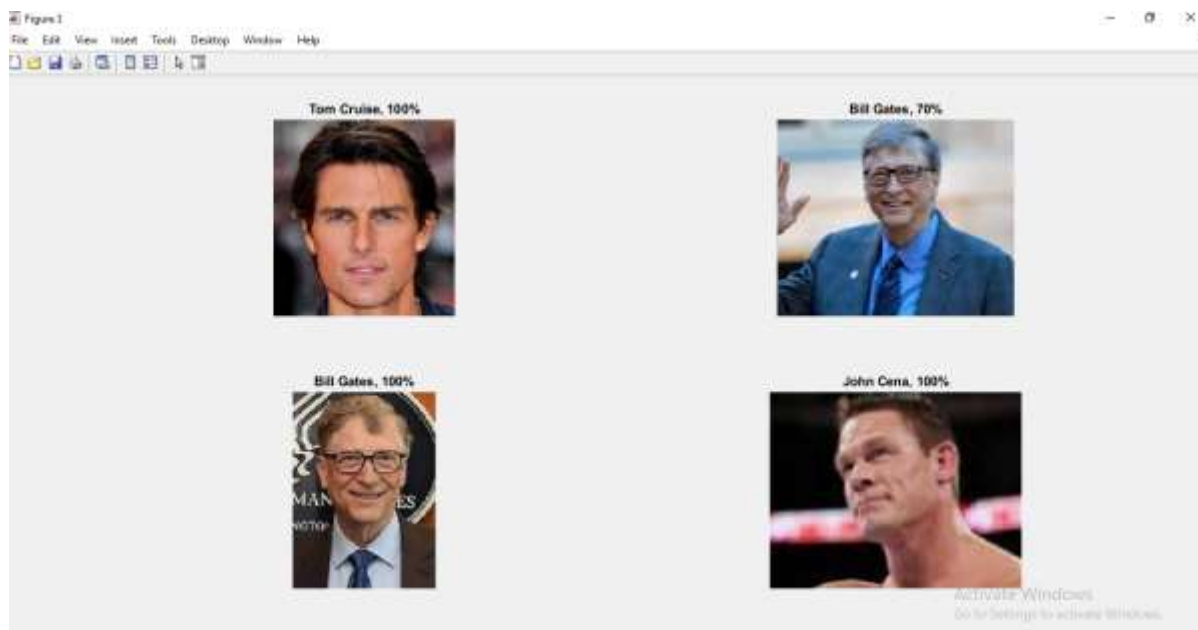


Fig.16. Training progress graph of SqueezeNet CNN





5997

Fig. 17. SqueezeNet CNN testing result

Table 3 shows the performance of both GoogleNet and SqueezeNet CNNs in terms of validation accuracy and training duration.

S. No	Network Type	Validation accuracy	Training time
1	GoogleNet CNN	88.24%	2 min 51sec
2	Squeeze Net CNN	100%	1 min 45sec

Table 3 Comparison between GoogleNet and SqueezeNet (during training process)

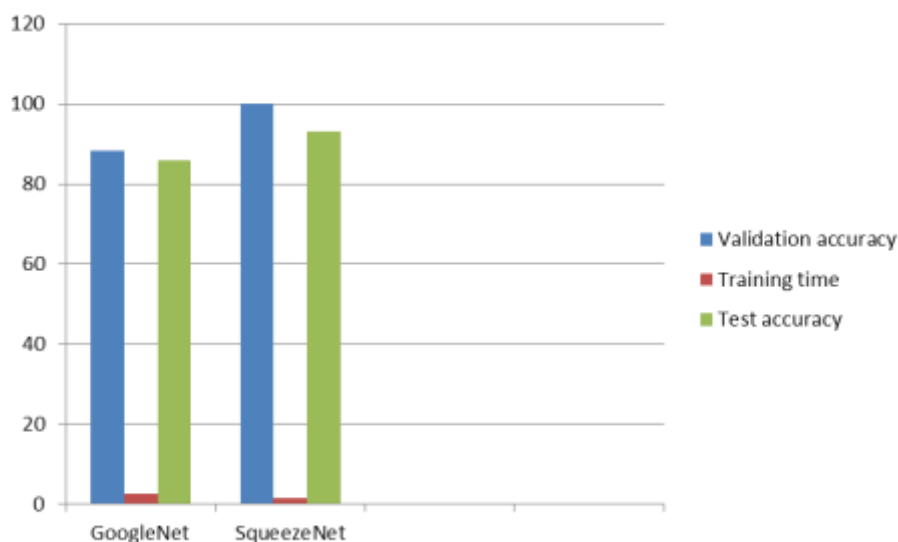


Fig. 18. Comparison between GoogleNet and SqueezeNet CNNs (during training process)

Table 4 shows the performance of both GoogleNet and SqueezeNet CNNs in terms of precision, recall, F1 score, and testing accuracy.



S. No	Network Type	Precision %	Recall %	F1-score %	Accuracy %
1	GoogleNet CNN	85.56	85.84	85.69	86
2	SqueezeNet CNN	92.23	91.38	91.87	93

Table 4 Comparisons between GoogleNet and SqueezeNet with precision, recall, f1 score and accuracy (during testing process)

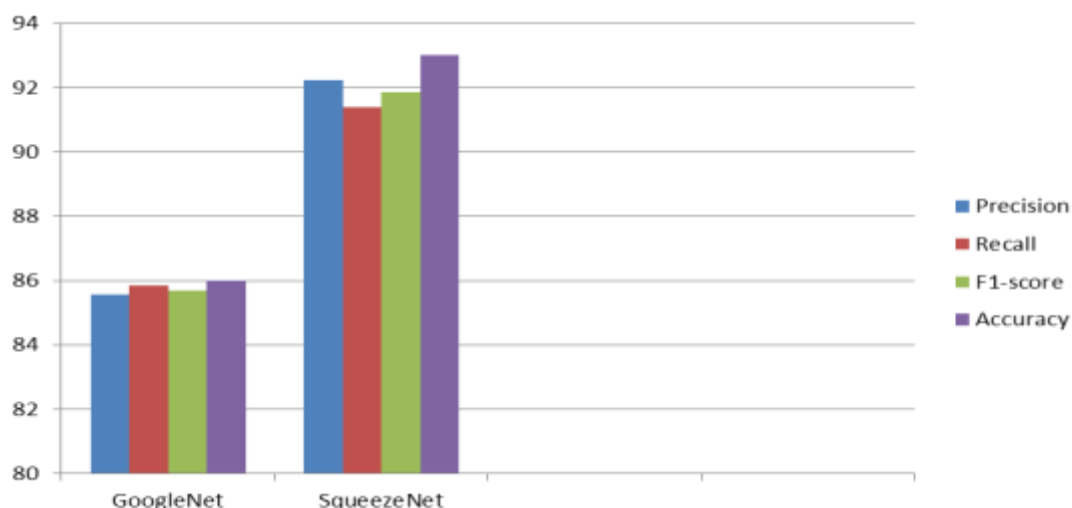


Fig. 19. Comparison between GoogleNet and SqueezeNet CNNs (during testing process)

Figures 18 and 19 illustrate that SqueezeNet CNN improves recognition accuracy while decreasing design complexity compared to GoogleNet CNN.

7. Conclusion and future work

This work explores the application of transfer learning on the SqueezeNet CNNs to do real-time face recognition. The performance of this model is compared with the GoogleNet CNN with a considerably deeper neural network that was trained exclusively for face recognition. The study demonstrates that SqueezeNet CNN performs better than GoogleNet CNN for face recognition tasks. In the future, we intend to extend the presented framework to solve more difficult recognition tasks, such as multiple face recognition and object recognition in general. In addition, we intend to conduct an exhaustive comparison of several deep learning models on large dataset for face recognition.

References

[1] Wang M., Deng W. "Deep face recognition: A survey", *Neurocomputing*, 2021; Vol. 429, pp. 215-244.

[2] Ding C., Tao D. "Trunk-branch ensemble convolutional neural networks for video-based face recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017; 40(4), pp.1002-1014.

[3] Parkhi O. M., Vedaldi A., Zisserman A. "Deep face Recognition". In *British Machine Vision Conference*, 2017; Vol. 1, pp.1-12.

[4] Yang J., Ren P., Zhang D, Chen D., et al. "Neural aggregation network for video face recognition". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017; pp.5216-5225.

[5] Iandola F. N., S. Han S, Moskewicz M.W., Ashraf K. "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size". *5th International Conference on Learning Representations*, 2017, pp.1-13.

[6] Zhang X., Zhou X., Lin M., Sun J. "Shufflenet: An extremely efficient convolutional neural network for mobile devices". *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017; pp.1-9.



- [7] Ma N., Zhang X., Zheng H. T., Sun J. "Shufflenet v2: Practical guidelines for efficient CNN architecture design". European Conference on Computer Vision, 2018; pp.122-138.
- [8] Sandler M., Howard A, Zhu M, Zhmoginov A, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". IEEE Conference on Computer Vision and Pattern Recognition, 2018; pp. 4510–4520.
- [9] Yu Z., Dong Y., Cheng J., Sun M. "Research on Face Recognition Classification Based on Improved GoogleNet". Security and Communication Networks, 2022; pp.1-6.
- [10] Omotosho L.O, Ogundoyin I.K., Oyeniyi J O. "A real time face recognition system using alexnet deep convolutional network transfer learning model". Journal of engineering studies and research, 2021; Vol. 27, pp.82-88.
- [11] Gholamalinezhad H., Khosravi H. "Pooling methods in deep neural networks, a review". Computer Vision and Pattern Recognition, 2020; pp.1-16.
- [12] Sun M., Song Z., Jiang X., Jiang X., Pan J., et al. "Learning pooling for convolutional neural network". Neurocomputing. 2017; Vol.224, pp.96-104.
- [13] Deng Y., Li D; Xie X., Lam K M., et al. "Partially occluded face completion and recognition". IEEE International Conference on Image Processing (ICIP), 2009; pp.4145–4148.
- [14] Geron A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow", 2nd Edition. O'Reilly Media, Inc., 2019.
- [15] Kukačka J., Golkov V., Cremers D. "Regularization for deep learning: A taxonomy". International Conference on Learning Representations, 2017; pp. 1-24.
- [16] Goodfellow I., Bengio Y., Courville A. "Deep Learning". MIT Press, 2016; pp.241–243.
- [17] Liu W., Zhou L, Chen J. "Face Recognition Based on Lightweight Convolutional Neural Networks". Information, 2021; Vol. 12, pp.1-18.
- [18] Szeged C., Liu W., Jia Y., Sermanet P., et al. "Going deeper with convolutions". IEEE Conference on Computer Vision and Pattern recognition. 2015; pp.1-12.
- [19] Simonyan K., Zisserman A. "Very Deep Convolutional Networks for Large-Scale Image Recognition". Computer Vision and Pattern Recognition., 2015; pp.1-14.

Authors Profile



Mr. P.Sankara Rao currently working as Assistant Professor at Raghu Engineering College, Visakhapatnam, India. He received his B.Tech degree from JNTUK University in 2009. He received his M.Tech degree from JNTUK University in 2012. He is doing Ph.D. in GIET university Odisha. He is He is a life member of the ISRD and IAENG. He has 10 years of teaching experience. He has published several national and international research papers in various journals and conferences. His research interest involves signal processing, image processing, Deep learning and computer vision and applications



Mr. K. Amarnadh is currently an undergraduate student pursuing B.Tech at Raghu Engineering College, Visakhapatnam, India. His research interest involves image processing, and computer vision applications.





Mr. J. Anu is currently an undergraduate student pursuing B.Tech at Raghu Engineering College, Visakhapatnam, India. Her research interest involves image processing, and computer vision applications.



Mr. K.V.S.Sanjay is is currently an undergraduate student pursuing B.Tech at Raghu Engineering College, Visakhapatnam, India. His research interest involves image processing, and computer vision applications.



Mr. M. Pavan Kalyan is currently an undergraduate student pursuing B.Tech at Raghu Engineering College, Visakhapatnam, India. His research interest involves image processing, and computer vision applications.



Mr. J. Sunny is currently an undergraduate student pursuing B.Tech at Raghu Engineering College, Visakhapatnam, India. His research interest involves image processing, and computer vision applications.

6000

