IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 January 2023, accepted 23 February 2023, date of publication 27 February 2023, date of current version 2 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3249787

RESEARCH ARTICLE

Handwritten Gujarati Numerals Classification Based on Deep Convolution Neural Networks Using Transfer Learning Scenarios

PARTH GOEL[1,2] AND AMIT GANATRA[3,4], (Senior Member, IEEE)

[1]Computer Science and Engineering Department, Devang Patel Institute of Advance Technology and Research (DEPSTAR), Faculty of Technology and Engineering, Charotar University of Science and Technology (CHARUSAT), Changa, Anand 388421, India
[2]U & P U Patel Department of Computer Engineering, Chandubhai S. Patel Institute of Technology (CSPIT), Faculty of Technology and Engineering, Charotar University of Science and Technology (CHARUSAT), Changa, Anand 388421, India
[3]Computer Science and Engineering Department, Faculty of Engineering and Technology, Parul University (PU), Vadodara, Waghodiava 391760, India
[4]Faculty of Technology and Engineering, Charotar University of Science and Technology (CHARUSAT), Changa, Anand 388421, India

Corresponding author: Parth Goel (parthgoel.ce@charusat.ac.in)

This work was supported in part by the Anita Devang Patel Ipcowala Center of Excellence in Artificial Intelligence (ADPICoE-AI), and in part by the Charotar University of Science and Technology (CHARUSAT).

ABSTRACT In recent years, handwritten numeral classification has achieved remarkable attention in the field of computer vision. Handwritten numbers are difficult to recognize due to the different writing styles of individuals. In a multilingual country like India, negligible research attempts have been carried out for handwritten Gujarati numerals recognition using deep learning techniques compared to the other regional scripts. The Gujarati digit dataset is not available publicly and deep learning requires a large amount of labeled data for the training of the models. If the number of annotated data is not sufficient enough to train Convolutional Neural Networks (CNN) from the scratch, transfer learning can be applied. However, the issue arises by using transfer learning is that how deep to fine-tune the pre-trained convolutional neural network while training the target model. In this paper, we addressed these problems using three deep transfer learning scenarios to classify handwritten Gujarati numerals from the images of zero to nine. We presented transfer learning scenarios using ten pre-trained CNN architectures including LeNet, VGG16, InceptionV3, ResNet50, Xception, ResNet101, MobileNet, MobileNetV2, DenseNet169 and EfficientNetV2S to find the best performing model by freezing and fine-tuning the weight parameters. We implemented the pre-trained models using a self-created handwritten Gujarati digit dataset with 8000 images of zero to nine digits with data augmentation. Exhaustive experiments are performed using various performance evaluation matrices. EfficientNetV2S model showed promising results among all the models including three transfer learning scenarios and achieved 98.39% training accuracy, 97.92% testing accuracy, 97.69% f1-score, and 97.15% AUC. Our handwritten Gujarati digit dataset is available on https://github.com/Parth-Goel/gujarati-handwritten-digit-dataset/.

INDEX TERMS Convolutional neural networks, Gujarati numerals, handwritten Gujarati digit dataset, classification, transfer learning, deep learning.

I. INTRODUCTION

India is one of the most linguistically diverse nations in the world with 28 states and 8 Union territories. Each state has its unique regional language including writing scripts. Gujarat is one of the states of India which is situated in the western region and Gujarati is the language of this state. The Gujarati language is derived from the Devanagari family of languages and it is spoken by more than 65 million people across the world [1]. Today, every organization including government or private sectors aimed for paperless work. However, banks, post offices, and any government or non-government offices

The associate editor coordinating the review of this manuscript and approving it for publication was Amin Zehtabian.

use to collect handwritten various paper documents which are generally written in regional languages. In the digital era, there is a requirement to convert these paper documents into a computer-readable format which leads to the development of Optical Character Recognition (OCR). OCR is a method to convert handwritten or printed text from electronic versions of documents into machine-readable text documents.

Gujarati OCR consists of Gujarati characters and numbers recognition. Our research work focuses on handwritten Gujarati numerals classification. Handwritten number recognition is a challenging task compared to printed documents because handwritten numbers vary depending on a person's writing style including the different sizes, curves, and thickness of the number. Handwritten number recognition is used in various applications for reading numbers from bank cheques, vehicle number plates, postal codes, houses, and any scanned forms. Moreover, handwritten number recognition is a primary task to read two or more digit numbers in a sequence. Significant research work has already been done for English [2], [3], Chinese [4], Arabic [5] and other Indian languages such as Bangla [6], [7], [8], Hindi [5], Kannada [9], Assamese [10], etc. Researchers also put effort for Gujarati digit or character classification using traditional machine learning techniques. However, very limited research work has been reported for Gujarati numerals classification using deep learning. Furthermore, to the best of our knowledge, authors have created their own Gujarati handwritten digit dataset for their research work but it is not provided publicly.

In recent years, deep learning has achieved significant success in the research field of artificial intelligence [11]. Deep learning algorithms are data-dependent and domain-specific. It also requires lots of data to train the algorithms because deep learning models learn the hidden characteristics of the data without hand-crafted features of the data. However, it is difficult and expensive to acquire applications' specific data and specifically labeled data. To reduce data dependency and to take the maximum utilization of existing data, transfer learning is the most effective technique. Transfer learning is the method for improving the performance of a new learning task in a new domain by utilizing the knowledge from the existing but related domain [12]. It is also referred to as transferability between the domains. In transfer learning, new domain, new task, existing domain, and existing task are referred to as target domain, target task, the source domain, and source task respectively. Typically, the transfer learning strategy reuses the trained models to acquire knowledge and reduces the development time of the target model substantially by avoiding the training from scratch. However, transfer learning is significantly dependent on transferability between the domains and the size of the target domain. In our study, the target task is to classify handwritten Gujarati numerals classification from the images which is also referred to as an image classification task. Image classification is performed using Convolutional Neural Networks (CNN) in deep learning. CNN architecture consists hierarchical representation of layers including convolution layers, pooling layers, and fully connected layers. Initial layers of CNN represent low-level features while higher layers detect abstract features which are specific to objects. Generally, low-level features are not transferred during the fine-tuning process in transfer learning because low-level features identify edges and textures. Thus it is important to determine the effective fine-tuning strategy for deep CNN to perform our target task i.e. handwritten Gujarati digit classification.

In this paper, we implemented the classification of handwritten Gujarati zero to nine digits using three transfer learning approaches to get the best performance up to the maximum extent. We utilized the ten pre-trained networks which include LeNet [13], VGG16 [14], Inceptionv3 [15], ResNet50 [16], ResNet101 [16], Xception [17], MobileNet [18], MobileNetV2 [19], DenseNet169 [20] and EfficientNetV2S [21]. In the first approach, the whole pre-trained model was used as a feature extractor and the feature vector of the last layer was trained with a linear Support Vector Machine (SVM) classifier and softmax classifier. We added two new fully connected layers after the last convolution layer of the pre-trained network in the second strategy and fine-tuned the newly adapted layers while freezing the remaining layers. In the third scenario, the pre-trained network was divided from the middle to take an advantage of weight parameters learning by the mid-level to high-level convolutional layers. We froze the layers of the first portion and fine-tuned the second portion with fully-connected layers. Usually, the pre-trained models are trained on the ImageNet dataset and it is considered as the de-facto database for transfer learning. ImageNet dataset consists of millions of images with 1000 classes. However, we also considered the MNIST dataset as a source domain with the LeNet model to investigate the effect of choosing another dataset than ImageNet.

The main contributions of this paper are summarized in the following:

- We have created a pioneer handwritten Gujarati numerals dataset from zero to nine with 800 images of each digit and provided this dataset publicly. Moreover, the process is also discussed to create this dataset.
- To investigate how much deep to train the pre-trained convolutional neural network using transfer learning in the proposed framework, we applied three transfer learning scenarios on ten pre-trained CNNs (LeNet, VGG16, ResNet50, ResNet100, InceptionV3, Xception, MobileNet, MobileNetV2, DenseNet169 and EfficientNetV2S) to show the effect of the different transfer learning strategies on the performance of the model and to get the best classification performance achievable to the maximum extent.
- Evaluation measures like training accuracy, testing accuracy, precision, recall, f1-score, total parameters, trainable parameters, CPU inference time, and training time were considered to evaluate the performance of individual pre-trained models with each transfer learning scenario.

- Finally, an overall comparison of three transfer learning scenarios was assessed and further performance analysis of the best performing model was presented using a confusion matrix.

The rest of the paper is organized as follows: Section II deliberates an overview of the related work. Section III explains the proposed framework using transfer learning scenarios with the pre-trained CNN architectures in detail. The process of self-created handwritten Gujarati numerals is demonstrated in section IV. Experimental analysis is discussed in section V along with a comparison using evaluation measures. Finally, section VI summarizes the paper with the conclusion and future directions.

II. RELATED WORK

In recent years, deep learning algorithms are applied in various computer vision applications. This paper presents handwritten Gujarati digit classification from images as a computer vision task. This section briefly discusses the research efforts addressed in the handwritten number and character recognition using traditional machine learning approaches and deep learning based approaches.

A. TRADITIONAL MACHINE LEARNING APPROACHES

Antani and Agnihotri attempted the first research study for Gujarati optical character recognition in the year 1999 using traditional machine learning techniques. They obtained digital images of Gujarati characters from the internet and scanned images of printed Gujarati text. They performed classification using the k–Nearest Neighbor (KNN) classifier with the Euclidean Minimum Distance and minimal hamming distance classifier (MHD). They achieved 67% and 48% accuracy with KNN and MHD classifiers respectively [22]. Dholakia et al. proposed a zone detection algorithm from the images of printed Gujarati characters. The algorithm computed the slopes of all imaginary lines that connect the top left and bottom right points of connected components which was applied to the three documents and extracted 20 lines correctly [23]. Dholakia et al. applied Daubechies D4 wavelet coefficients to extract the features from printed Gujarati characters. These feature vectors were used with the general regression neural network (GRNN) classifier and obtained 96% accuracy [24]. The structural feature extraction approach was introduced to classify printed Gujarati characters by Goswami and Mitra [25]. They evaluated their approach on 4000 different printed Gujarati characters and showed 92.7% accuracy. In [26], the authors employed a binary multiple kernels learning-based classification algorithm for Gujarati and Bangla character recognition. The main limitation of these works is that authors have used printed Gujarati digits for classification. The printed digits have the same types of writing style, curves, and strokes compared to handwritten numerals. In these methods, the effectiveness of features and classification performance is dependent on digit font type and hence these methods may work better for a few related font types of digit but they may give a worse performance for other types of digits. Moreover, they have used a limited number of samples for training with traditional machine learning approaches.

The multi-layered feed-forward neural network was proposed for Gujarati handwritten digit classification by Desai [27]. For each digit, 278 images of Gujarati numbers were used for experimentation. These digits were pre-processed by thinning and skew correction. Two diagonal, vertical, and horizontal projection profiles were applied to extract the features. This work obtained 82% accuracy for Gujarati handwritten digit classification. Maloo and Kale applied affine invariant moments to extract the features from the sample images. They used 80 images per Gujarati digit and these images were pre-processed using morphological operations. Then the features were fed to the Support Vector Machine (SVM) classifier for handwritten Gujarati numeral classification and reported 91% accuracy [28]. Baheti et al. compared the KNN classifier and PCA by using the Euclidean distance equation to recognize the Gujarati handwritten digits [29]. They collected 80 samples of each digit and used affine invariant moments for feature extraction. They achieved 90% and 84% accuracy for the KNN classifier and PCA respectively. Artificial Neural Networks, Support Vector Machine, and Naive Bayes classifiers were employed by Sharma et al. for handwritten Gujarati digit recognition. They have extracted features using chain code-based, zone-based, and projection profile-based methods. They achieved the highest accuracy using a support vector machine and reported 98.75% [30]. Support Vector Machine was examined by Naik et al. using radial basis, polynomial, and linear function kernels. They achieved an accuracy of 92.60 % for linear kernels, 95.80% for polynomial kernels, and 93.80% for RBF kernel on the handwritten Gujarati numeral dataset. [31]. Bharvad et al. implemented the Support Vector Machine (SVM) classifier and compared it with machine learning supervised classifiers for handwritten Gujarati digits. They obtained a maximum accuracy of 92% [32]. These works have utilized the traditional machine learning approaches which performed the classification using handcrafted features with few numbers of digits. Thus, these approaches are less effective to generalize more into real-world applications.

B. DEEP LEARNING APPROACHES

Traditional machine-leaning methodologies have been used to investigate a significant amount of work for Gujarati handwritten characters and digits. Very limited research efforts have been applied to the study of handwritten Gujarati numerals using deep learning approaches. Hence we have also considered other languages' digit classification work than Gujarati digit classification using deep transfer learning methods. Deep learning minimizes or removes the need for feature extraction as it is required in traditional machine-leaning. In addition to the typical conventional machine learning approaches, recent research has emphasized the use of a

pre-trained Convolutional Neural Network (CNN) for handwritten digit classification using transfer learning.

In [33] by Shopon et al., handwritten Bangla digits were recognized using a pre-trained autoencoder and a CNN. Pramanik et al. demonstrated the use of transfer learning with the pre-trained Convolutional Neural Network (CNN) architectures by combining the data from handwritten numbers in Bangla and Oriya. They showed improved results with AlexNet and VGG16 pre-trained networks [34]. Zunair et al. utilized a VGG16 pre-trained model to classify handwritten Bengali digits. They reported the highest accuracy of 97.09% [35]. An extensive review has been conducted in [36] for offline Bengali handwritten digit recognition and presented a comprehensive insight of state-of-the-art datasets and approaches based on image processing, traditional Machine Learning (ML) and Deep Learning (DL) architectures including several real-life applications. Shukla and Desai proposed a deep learning approach for the recognition of Handwritten Gujarati Characters and Numerals. They obtained 82.15% test accuracy [37]. In [38], the authors employed a convolutional neural network to classify the handwritten Assamese digits and also compared it with VGG16 pre-trained networks. They have reported 93.02% testing accuracy. Limbachiya et al. applied transfer learning using five pre-trained networks including VGG16, InceptionV3, DenseNet, Nasnet, and MobileNet on handwritten Gujarati alphanumeric characters. They obtained the highest accuracy of 97% using the MobileNet pre-trained network [1]. Goel et al. utilized a transfer learning approach to classify Gujarati digits using VGG16, VGG19, ResNet50, ResNet101, and EfficentNet pre-trained models. They used 250 images of each digit in experiments and reported 96.5% testing accuracy by the EfficientNet model [39].

In these approaches, the authors have employed transfer learning with deep learning pre-trained CNN with only one scenario which does not necessarily provide the best performance of the model. Therefore, it is crucial to perform extensive experiments with various deep transfer learning scenarios to ensure the applicability of features and classification approaches for Gujarati handwritten digits. Furthermore, the authors have used a few samples of each digit without data augmentation which is very less for deep transfer learning to generalize the model. Additionally, the authors who contributed their research in Gujarati handwritten digit classification have not provided their Gujarati handwritten digit dataset publicly to reuse for further research work.

III. METHODOLOGY

In this paper, we propose Gujarati handwritten numerals classification using the CNNs-based transfer learning scenarios to perform classification of zero to nine digits. We investigate various pre-trained CNN models using transfer learning scenarios in order to acquire the optimal classification performance achievable to the maximum extent. In this section, approaches used for Gujarati handwritten digit classification are explained in detail.

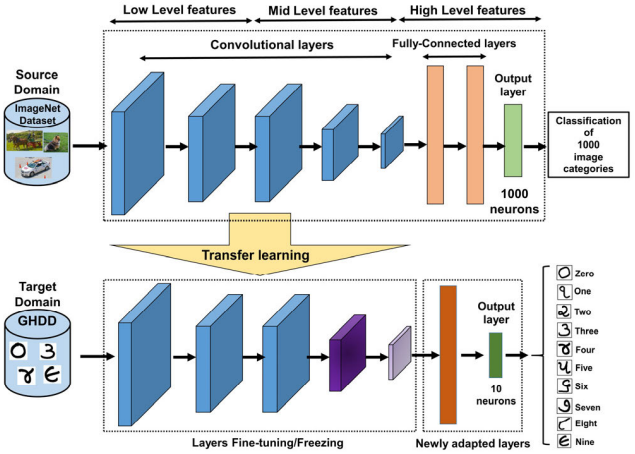

FIGURE 1. The pipeline of Transfer Learning from ImageNet pre-trained CNN models to Gujarati handwritten digit classification.

A. TRANSFER LEARNING

Machine learning or deep learning algorithms are problem-specific and data-dependent. Whenever the domain changes, these algorithms need to be retrained in the new domain. However, these algorithms can be retrained using transfer learning strategies that use knowledge from the source domain to improve the performance of the target domain for similar tasks. Thus, transfer learning tries to utilize existing models and information without recreating them from scratch. Therefore, it reduces the model development time and improves the model performance significantly. The source and the target domain are from the related feature space which is images. Hence, a homogeneous transfer learning setting is applied in our work based on Weiss et al. [40]. Moreover, our source task and target task are different. According to Pan and Yang [12], inductive transfer learning is used when the source and target tasks are distinct. Thus inductive transfer learning setting is employed in our work because the source task is the classification of various objects from images and the target task is the classification of Gujarati handwritten digits from 0 to 9 images.

The CNN architecture is comprised of various hierarchical layers of convolutional and max-pooling layers that culminate with one or more fully-connected layers. These layers are used to learn complicated patterns from large amounts of data. Figure 1 illustrates the proposed pipeline of Transfer Learning from ImageNet pre-trained CNN models to Gujarati handwritten digit classification. The pre-trained models used the ImageNet dataset as the source domain and the proposed model used the Gujarati Handwritten digit dataset (GHDD) as the target domain. ImageNet dataset has 1000 categories and GHDD has 10 categories. Initial layers of CNN attempt to learn low-level features such as edges and texture of the object and deeper layers of CNN aim to capture more intricate and abstract level features of the object. Transfer learning utilizes the parameters from a well-trained model from the source domain to assist in training a model in the target domain. If the two domains are related, then certain low-level CNN features

can be reused, while high-level parameters can be fine-tuned or frozen using transfer learning strategies by adding new layers at the end of CNN.

In this work, we present three scenarios to train the Convolutional Neural Network for Gujarati Handwritten digit classification to get the best results up to the maximum extent by considering the size and similarity of the source and target domain. The three approaches are based on transfer learning which includes the pre-trained CNN models as fixed feature extractors and fine-tuning the few last layers. These transfer learning approaches are presented in figure 2. We considered ten popular pre-trained CNN models namely LeNet, VGG16, InceptionV3, ResNet50, Xception, ResNet101, MobileNet, MobileNetV2, DenseNet169 and EfficientNetV2S. The architecture of LeNet consists of 8 layers including 3 convolution layers, 2 subsampling layers, 2 fully connected layers of the size 120 and 80 neurons, and one output layer. We have trained the LeNet model using the MNIST dataset to get the trained weight file for transfer learning to Gujarati handwritten digit classification. The InceptionV3 model has a depth of 189 layers which includes 94 convolution layers, 94 pooling or relu activation layers, and one output layer. VGG16 network has 16 layers in which 13 convolution layers, 2 fully connected layers, and one output layer. ResNet50 is 107 layers long which consists of 49 convolution layers, 57 pooling or relu activation layers, and one output layer. The Xception model includes 40 convolution layers, 5 pooling layers, 35 relu activation layers, and one output layer and the total depth of layers is 81. ResNet101 network has 209 layers including convolutional, relu and pooling layers. MobileNet and MobileNetV2 are shallow deep learning models having depth-wise and point-wise convolution with 55 and 105 layers respectively. DenseNet has three variations which include DesneNet121, DesneNet169 and DesneNet201. Similarly, EfficientNet has fifteen versions including EfficientNetV2. DenseNet169 and EfficientNetV2S are selected among all their variations based on the balance between model size and accuracy in our experiments. DensNet169 comprises four dense blocks which consist 164 convolutional layers and 4 other convolutional layers and one fully connected layer. EfficientNetV2S includes six blocks including 101 convolutional layers. These pre-trained CNN models are utilized as deep feature extractors as a model in the first approach and are presented in section (a) of figure 2. In section (b) of figure 2, the second approach is shown where the last fully connected layers are removed from the pre-trained CNN models and new fully-connected layers are adapted. Few convolution layers and fully connected layers are modified in the third approach and illustrated in section (c) of figure 2. These scenarios are explained in the following subsequent sections.

B. PRE-TRAINED CNN MODELS AS A FIXED FEATURE EXTRACTOR

In this approach, pre-trained convolution neural networks are used as a fixed feature extractor. Pre-trained CNNs are trained on the ImageNet dataset which has 1000 classes. Input images are fed into the pre-trained models directly. Trainable parameters are not fine-tuned or updated from the whole network except the last fully connected layer. These layers have extracted the features from the input images i.e. Gujarati handwritten digit dataset. Extracted features are sent for the training to the linear SVM and softmax classifier individually for the classification of Gujarati digits as shown in figure 2-(a). The advantage of this approach is that the network spends less time in training. Because convolution operations are computationally costly and while training the network these operations are performed multiple times. However, in this approach, all the layers of CNNs are kept frozen and the input images are only passed through the network once.

C. PRE-TRAINED CNN LAYERS AS A FIXED FEATURE EXTRACTOR

CNNs comprise multiple layers including convolutional layers, max-pooling layers, and fully connected layers that learn hierarchical representations of data. Among these layers, weight parameters of convolutional layers and fully connected layers are trainable during the training of the networks. Typically, convolutional layers are responsible to extract the features whereas the fully connected layers are mapped the extracted features into high-level object-specific features. The lower layers of the CNNs learn the basic features and the higher layers learn the specific features of the given object in the images. In this approach, convolutional layers of the pre-trained CNN models are utilized as a feature extractor and kept these layers frozen during the training. Fully connected layers from the pre-trained CNNs are removed and two new fully connected layers with the size of 512 and 256 neurons are added as depicted in figure 2-(b). In the last fully connected layers, the number of neurons is kept 10 instead of 1000 because there are 10 classes of Gujarati digits which is referred to as the output layer. Finally, these two fully connected layers and an output layer with a softmax classifier are trained with the Gujarati handwritten digit dataset for ten pre-trained CNNs separately.

D. FINE-TUNING CNN LAYERS

In this approach, pre-trained CNN models are trained with partial layers of the networks with the same architecture. To train all layers or partial layers of pre-trained CNN models which depend on the correlation between the target and source dataset. In our study, the similarity between the target domain and the source domain is not significant. Thus, this approach starts training with partially selected CNN layers, fully connected layers, and output layers. These selected CNN layers of the network learn mid to high-level representations of the target domain i.e. Gujarati handwritten digit dataset for the target task i.e. Gujarati handwritten digit classification. The last half of the CNN layers are chosen from the ten selected pre-trained models to train the weights parameters. Two fully connected layers with the size of 512 neurons

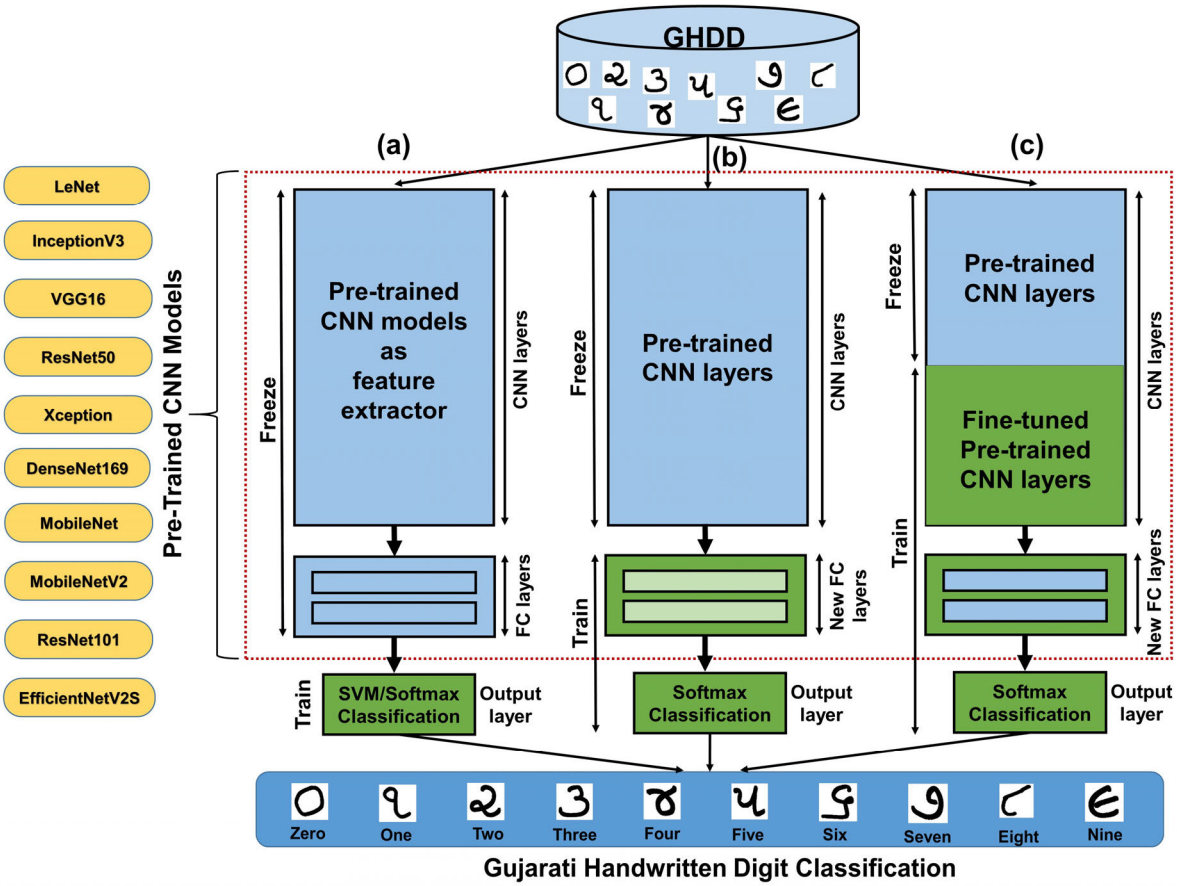

FIGURE 2. The proposed approaches for Gujarati handwritten digit classification; (a) pre-trained CNN models as feature extractor and output layer trained using SVM/Softmax classifier; (b) training by freezing the convolution layers, newly adapted fully connected layers with softmax classifier; (c) training by freezing partial convolution layers, fine-tuning partial convolution layers and newly adapted fully connected layers with softmax classifier.

and 256 neurons followed by an output layer of 10 neurons and a softmax classifier are replaced with the existing setup of pre-trained models. Finally, updated ten pre-trained CNN models are trained with the Gujarati handwritten digit dataset separately.

IV. DATASET INFORMATION

English handwritten digit dataset is widely available such as MNIST [11] and it is extensively utilized in research for digit classification. However, Gujarati handwritten digit dataset is not available publicly to the best of our knowledge. One of the substantial contributions of this study is the building of a dataset for handwritten Gujarati digit recognition. In this section, the dataset description is presented including dataset samples information, the collection process, and the pre-processing of the data.

A. DATA COLLECTION

Gujarati Handwritten digit dataset collection was a very difficult task due to the lack of availability of the dataset. A handwriting style is influenced by a variety of circumstances, including the writer's education, career, age, and writing pen. To accommodate these variations, the data samples were collected on blank A4 size white paper from different individuals of our college such as peons, sweepers, students, workers, drivers, teaching staff, and lab assistants. Figure 3 illustrates the sample pages of Gujarati numbers collected from various people.

One sample page was collected from 150 persons with different backgrounds. Each person wrote ten digits from zero to nine on a single A4-size white paper. Five to nine times each digit was written on a paper based on the writing style of individuals and the space on a page. It can be seen in figure 3. Black and blue ball pens with different size of tips and CD marker pens were used to write digits on paper. There were no conditions put on any individuals to write the digits. Thus maximum versatility can be achieved in the dataset including varying sizes of fonts. These input samples were scanned using a flatbed scanner at 300 dpi resolution and saved in greyscale images [27].

B. DATA PRE-PROCESSING

The pre-processing of data is required to enhance the quality of the dataset which increases the recognition rate of

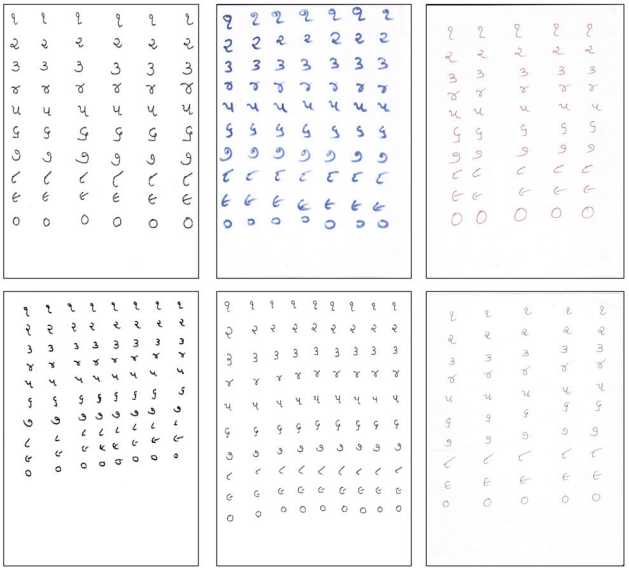

FIGURE 3. Sample pages of Gujarati handwritten digit samples from different individuals.

FIGURE 4. Subdivided digits which have individual strokes.

classification models. Following pre-processing steps were applied to convert the page samples into individual digits.

1) DIGIT SEGMENTATION

Segmentation was applied to get individual digit images from the input sample. Firstly, Gaussian blur is applied with a 5 × 5 kernel to reduce high-frequency noise and to make the contour detection process more accurate. Secondly, thresholding is employed to convert input images into black-and-white images from greyscale images. Thresholding makes the segmentation process easier because it allows us to separate the background and the foreground. Lastly, the contour method is applied to get all possible contours from input samples. However, the contour segmentation method was not able to segment subdivided digits that have individual strokes. There is only digit nine in the Gujarati digits which is not continuous. It shows in figure 4. This method correctly segments continuous digits from zero to eight as these digits do not have individual strokes. To address this issue, we investigated using the morphological operation of dilation to simply dilate the digit to obtain linked strokes. After dilatation, Erosion was performed to restore the digit's original width. This resulted in the majority of nine digits in completely continuous digits that can be segmented using the contour segmentation method. Small sizes of contours are eliminated to reduce the noise from segmented images.

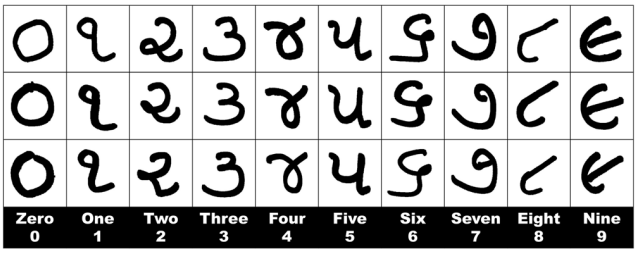

FIGURE 5. Examples of Gujarati handwritten numerals from the dataset.

2) DIGIT RESIZING

We obtained samples of varied sizes and fonts because there were no limitations in writing the numerals. To ensure consistency, all digit images were resized into 256 × 256 pixels.

3) NOISE ELIMINATION AND THINNING

While scanning the page or different writing styles of individuals, salt and pepper noise was produced in the segmented images. We applied morphological operations dilation and erosion. Dilation was utilized to remove the noise and smooth the boundary of digits from segmented images. Erosion was used to thin the width of the digit. A few noisy digit samples are removed from the dataset and finally produced a balance dataset of 800 samples of each Gujarati digit with a resolution of 256 × 256. The total number of images is 8000 for ten classes. The dataset is available online at https://github.com/Parth-Goel/gujarati-handwritten-digit-dataset/. Examples of Gujarati handwritten numerals are presented in figure 5.

V. EXPERIMENTAL ANALYSIS

In this section, experimental analysis is carried out using transfer learning on the Gujarati handwritten digit dataset. Moreover, three transfer learning scenarios with ten pre-trained models are discussed, and also shown the effects of them to choose the best model.

A. PERFORMANCE EVALUATION MATRICES

The performance of the models has been evaluated using various evaluation measures which include training accuracy, testing accuracy, precision, recall, f1-score, AUC, number of parameters, trainable parameters, CPU inference time, and training time on target data. Accuracy, precision, recall, f1-score and AUC have been calculated using equations (1) – (5) [41]. True Positive (TP) represents the number of correctly classified digits. False Positive (FP) indicates the number of misclassified digits. False Negative (FN) shows the number of digits misclassified as correct digits however they are not. True Negative (TN) represents the number of correctly classified negative digits that are actually other classes. AUC (Area Under the Curve) measures

the degree of separability which shows the capability of the model for distinguishing among the classes. It is calculated by taking an average of recall and specificity. In equation 5, the first term represents recall and the second term indicates specificity. Training accuracy indicates the accuracy of the data which are used for training the model. Testing accuracy means that the trained model evaluates the performance on unseen data but the distribution is the same. Total parameters represent the trainable and non-trainable parameters of the pre-trained networks. Trainable parameters are used to train the model which is actually updated while training the models to learn the features. CPU inference time is the total time takes for forward propagation of the trained model to classify the image. Training time is the total time taken for training the model including forward pass and backward pass.

$$Accuracy = (TP + TN)/(TP + FP + FN + TN) \quad (1)$$

$$Precision(P) = TP/(TP + FP) \quad (2)$$

$$Recall(R) = TP/(TP + FN) \quad (3)$$

$$F1 - Score = (2 \times P \times R)/(P + R) \quad (4)$$

$$AUC = [\{TP/(TP + FN)\} + \{TN/(TN + FP)\}]/2 \quad (5)$$

B. EXPERIMENTAL SETUP

Experiments were performed on a machine with an Nvidia RTX A4000 (16GB) GPU, 64 GB RAM, 4 TB hard disk, Intel Xeon Silver 4210R CPU 2.4GHz processor, and 64-bit Windows operating system. The dataset was pre-processed using Python 3.7 and OpenCV library. The proposed pre-trained models were developed using Python 3.7, TensorFlow 2.8.2, Keras 2.8.0, and Scikit-Learn 1.0.2.

The dataset has been divided into fixed sets of train set and test set and split into 70:30 for the training and testing data respectively. The train dataset folder contains 560 images and the test dataset folder contains 240 images out of 800 in total images. Images were resized to 128 × 128 for experiments. During the training, the learning rate was set to 0.0001, the epoch was 10, the momentum was 0.9, and the batch size was 64. There is an equal number of samples for each class in the training set and testing set. Thus there is no mechanism applied for the data balancing as our dataset is balanced. We have run all the models five times and average values are reported in the result tables. We have set dropout to 0.5 which was used to handle the overfitting problem. We have also used the ReduceLROnPlateau method with early stopping which decreases the likelihood of an overfitting problem by reducing the learning rate when it stops progressing and kept a 0.00001 minimum learning rate and patience value was 3. Adam optimizer with backpropagation and categorical cross-entropy loss methods were employed for training the models.

Data augmentation techniques were applied to artificially generate the digit samples and increase the size of the dataset. This approach helped to reduce the over-fitting during the model training and increase the generalization ability of the model. We have employed several data augmentation techniques with handwritten Gujarati digits which include rotation (left and right orientations), zooming (in and out), shifting (width and height), shearing and background color change. We used the same data augmentation techniques on the training set and test set which increase the size of the dataset by a factor of eight. This increases the size of our datasets by a factor of 8. We have not applied the horizontal and vertical flipping because these methods of data augmentation change the originality of writing pattern of digit. Figure 6 shows the sample output images of a digit two after applying data augmentation techniques.

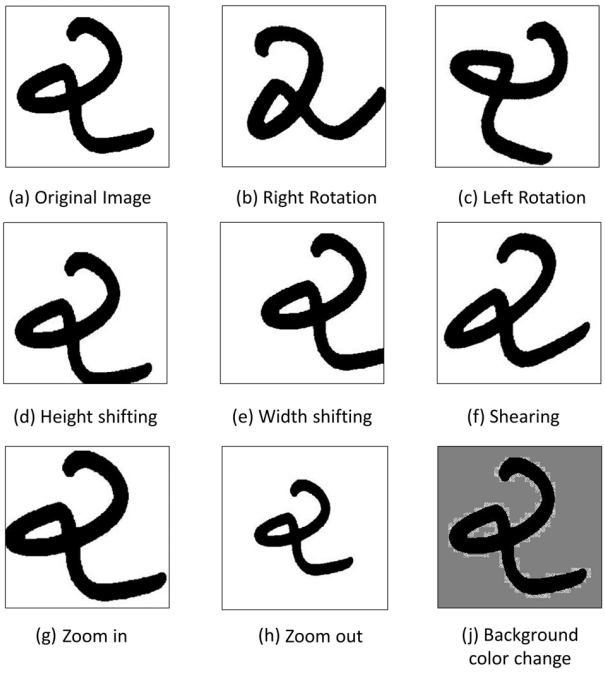

FIGURE 6. Sample images after applying data augmentation techniques.

1) RESULTS

The LeNet, VGG16, InceptionV3, ResNet50, Xception, ResNet101, MobileNet, MobileNetV2, DenseNet169 and EfficientNetV2S pre-trained models were employed to investigate three transfer learning scenarios using the Gujarati handwritten digit dataset. Generally, transfer learning is preferred to apply based on the similarity of the source domain and target domain which utilize the weights parameter of existing pre-trained models and train on the target domain in minimum time with the best performance. Thus, the LeNet model is considered to evaluate the performance of the Gujarati handwritten digit dataset. LeNet was pre-trained on the MNIST dataset and this model gained popularity in handwritten English digit classification. However, recent studies show that pre-trained models trained on the ImageNet dataset showed the best performance in various image classification tasks. The depth of these models is deep compared to the LeNet model. The VGG16, InceptionV3, ResNet50, Xception, ResNet101, MobileNet, MobileNetV2, DenseNet169

TABLE 1. Results of 1st scenario of transfer learning using SVM classifier - Pre-trained CNN models as a fixed feature extractor.

Parameters / Model Name	Training Accuracy (%)	Testing Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC (%)	Total parameters	Trainable parameters	CPU Inference time (ms)	Training time (minutes)	Model Size (MB)
MobileNet	90.71	88.75	87.25	90.45	88.82	88.24	3.3927M	163.850K	23.42	2.46	12.94
VGG16	91.61	89.17	89.45	92.42	90.90	89.25	14.7966M	81.930K	78.54	3.88	56.44
MobileNetV2	91.79	89.17	90.12	91.51	90.80	90.12	2. 4627M	204.810K	28.87	2.79	9.39
ResNet50	91.96	89.92	90.15	88.56	89.35	91.25	23. 9154M	327.690K	66.21	4.12	91.23
IncpetionV3	92.86	90.83	90.01	91.25	90.63	89.84	21.8847M	81.930K	53.46	3.93	83.48
Xception	93.75	90.83	89.25	91.34	90.26	91.12	21. 1891M	327.690K	119.84	5.45	80.83
EfficientNetV2S	93.46	91.25	90.35	91.82	91.07	90.50	20.5362M	204.810K	194.39	5.96	78.33
DenseNet169	92.68	91.67	89.95	91.25	90.60	90.25	12.9091M	266.250K	104.51	5.50	49.24
ResNet101	**93.93**	91.67	91.56	89.89	90.72	90.80	42.9858M	327.690K	95.65	4.95	163.97
LeNet	93.57	**93.33**	**92.56**	**95.45**	**93.98**	**91.23**	**1.6287M**	**0.850K**	**11.63**	**1.05**	**5.18**

TABLE 2. Results of 1st scenario of transfer learning using Softmax classifier - Pre-trained CNN models as a fixed feature extractor.

Parameters / Model Name	Training Accuracy (%)	Testing Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC (%)	Total parameters	Trainable parameters	CPU Inference time (ms)	Training time (minutes)	Model Size (MB)
MobileNet	88.04	89.17	86.83	89.25	88.02	88.01	3.3927M	163.850K	23.42	2.46	12.94
VGG16	91.79	90.00	91.52	90.20	90.85	89.43	14.7966M	81.930K	78.54	3.88	56.44
MobileNetV2	92.14	90.42	90.57	92.32	91.42	90.21	2.4627M	204.810K	28.87	2.79	9.39
ResNet50	92.50	91.25	90.24	89.87	90.05	90.53	23.9154M	327.690K	66.21	4.12	91.23
IncpetionV3	93.39	92.08	90.52	91.86	91.18	91.19	21.8847M	81.930K	53.46	3.93	83.48
ResNet101	93.75	92.08	91.35	90.45	90.89	91.45	42.9858M	327.690K	95.65	4.95	163.97
DenseNet169	93.04	92.50	90.36	91.24	90.79	90.40	12.9091M	266.250K	104.51	5.50	49.24
Xception	94.29	92.50	91.71	90.83	91.26	91.48	21.1891M	327.690K	119.84	5.45	80.83
EfficientNetV2S	93.64	92.92	91.45	92.72	92.08	91.74	20.5362M	204.810K	194.39	5.96	78.33
LeNet	**93.92**	**94.16**	**93.52**	**94.33**	**93.92**	**92.46**	**1.6287M**	**0.850K**	**11.63**	**1.05**	**5.18**

and EfficientNetV2S pre-trained models are considered to evaluate Gujarati handwritten digit classification task. These pre-trained models were trained on the ImageNet dataset and the weights file of the trained models was obtained using the Keras libraries. All ten models are evaluated as per the three transfer learning scenarios explained in the methodology section and the result analysis is explained in the subsequent parts.

In the first approach, the pre-trained CNN models were considered as feature extractors and kept freeze the configuration and parameters up to the second last layer of the models. The feature vector of the second last layer was fed into the linear SVM and softmax classifier individually for the handwritten Gujarati digit classification. Softmax classifier was used to analyze the role of the SVM classifier. Table 1 and Table 2 show the performance analysis of the ten pre-trained networks using the SVM classifier and softmax classifier respectively. It can be observed that the LeNet model outperformed among all ten models in this scenario. The softmax classifier achieved 93.92% training accuracy, 94.16% testing accuracy, 93.52% precision, 94.33% recall, 93.92% f1-score, and 92.46% AUC whereas linear SVM classifier obtained 93.57% training accuracy, 93.33% testing accuracy, 92.56% precision, 95.45% recall, 93.98% f1-score, and 91.23% AUC. The size of the feature vector of the LeNet model is 850 which is passed as an input to linear SVM and the softmax classifier which is very less compared to other models. Thus, this model took minimum time for training and inferencing which is 1.05 minutes and 11.63 ms respectively. MobileNet shows the lowest performance among all models by both of the classifiers. Tables 1 and 2 also reveal that the highest training time and inferencing time are taken by the EfficientNetV2S model which is 5.96 minutes and 194.39 ms respectively. The highest number of parameters, trainable parameters and model size are reported by ResNet101 in both Tables.

TABLE 3. Results of 2nd scenario of transfer learning - Pre-trained CNN layers as a fixed feature extractor.

Parameters / Model Name	Training Accuracy (%)	Testing Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC (%)	Total parameters	Trainable parameters	CPU Inference time (ms)	Training time (minutes)	Model Size (MB)
MobileNet	91.96	90.42	89.50	92.80	91.12	91.25	11.7519M	8.5230M	24.61	5.62	44.82
MobileNetV2	93.75	92.08	93.40	92.70	93.05	92.36	12.8781M	10.6201M	29.22	6.21	49.12
VGG16	93.21	92.92	92.10	93.70	92.89	92.46	19.0434M	4.3287M	79.11	7.55	72.64
DenseNet169	94.82	94.58	93.60	95.10	94.34	94.25	26.4087M	13.7658M	106.24	9.84	100.74
IncpetionV3	95.36	95.00	96.20	95.40	95.80	94.75	26.1314M	4.3287M	53.86	6.56	99.68
ResNet50	94.64	95.00	96.20	93.80	94.98	94.50	40.4993M	16.9116M	67.31	8.52	154.49
LeNet	96.25	95.41	96.40	95.80	96.10	94.60	**1.6285M**	**1.6258M**	**11.67**	**2.21**	**6.21**
ResNet101	96.07	95.83	94.80	95.80	95.30	94.70	59.5698M	16.9116M	95.81	9.38	227.24
Xception	97.86	96.67	95.30	97.20	96.24	96.75	37.7731M	16.9116M	120.43	10.53	144.09
EfficientNetV2S	**98.39**	**97.92**	**96.80**	**98.60**	**97.69**	**97.15**	30.9515M	10.6201M	194.82	12.34	118.07

TABLE 4. Results of 3rd scenario of transfer learning – Fine-tuning CNN layers.

Parameters / Model Name	Training Accuracy (%)	Testing Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC (%)	Total parameters	Trainable parameters	CPU Inference time (ms)	Training time (minutes)	Model Size (MB)
MobileNet	91.07	89.58	92.80	90.40	91.58	92.10	11.7519M	11.4608M	25.42	11.25	44.82
MobileNetV2	93.75	91.67	93.85	92.56	93.20	92.56	12.8781M	12.7132M	30.24	11.84	49.12
VGG16	92.50	91.67	91.30	92.20	91.75	92.40	19.0434M	17.3079M	79.88	13.29	72.64
ResNet50	94.29	93.33	95.70	93.50	94.59	93.95	40.4993M	38.9962M	67.54	15.81	154.49
DenseNet169	94.29	93.75	94.50	92.80	93.64	92.80	26.4087M	23.1951M	106.81	17.56	100.74
LeNet	95.89	94.58	93.56	92.80	93.18	93.40	**1.6287M**	**1.6282M**	**11.87**	**4.87**	**6.21**
IncpetionV3	95.00	94.58	93.25	94.20	93.72	93.24	26.1314M	21.6956M	54.26	11.46	99.68
ResNet101	96.25	95.00	94.80	93.50	94.15	93.78	59.5698M	49.7871M	95.89	17.42	227.24
Xception	96.79	95.83	94.65	95.80	95.22	94.25	37.7731M	31.3698M	121.21	20.41	144.09
EfficientNetV2S	**97.32**	**96.67**	**95.40**	**96.10**	**95.75**	**94.72**	30.9515M	26.3078M	195.22	22.63	118.07

In the second approach, the pre-trained CNN models were kept freeze the configuration and parameters up to the last convolution layer and fine-tuned the newly adapted fully connected layers and output layer. The results of this approach are presented in Table 3. It can be seen that the EfficientNetV2S model achieved the highest testing accuracy 97.92%, f1-score 97.69%, and AUC 97.15%. This model took 12.34 minutes and 194.82 ms to train and infer which is the maximum among all models. ResNet101 obtained the highest value in total parameters, trainable parameters and model size which are almost double of EfficientNetV2S. The MobileNet model reported the lowest performance for all evaluation measures compared to the other nine models.

In the third approach, the pre-trained model was chopped into half of the network and the first half network was kept frozen and the later part of the network was fine-tuned. LeNet model was chopped from the 2nd convolutional layer. InceptionV3 model was broken down from the 6th mixed block. VGG16 model was fine-tuned from the 4th CNN block of the network. ResNet50 and ResNet101 models were cut in half of the network from the 4th CNN block. The Xception model was divided into two parts from the 7th block of the network. MobileNet and MobileNetV2 were fine-tuned from the 43rd and 72nd layer respectively. DenseNet169 was cut from the 4th block of the network. Lastly, EfficientNetV2S was divided from the 5th block of the model. Table 4 reveals the performance analysis of the pre-trained models using this approach. It can be observed that the EfficientNetV2S model shows better results than the others. The training accuracy, testing accuracy, f1-score, and AUC are 97.32%, 96.67%, 95.75%, and 94.72% respectively. The highest total parameters, trainable parameters and model size are reported 59.5698M, 49.7871 and 227.24M respectively by the ResNet101 model. The lowest performance was obtained by the MobileNet model with 89.58% testing accuracy, 90.40% f1-score and 92.10% AUC.

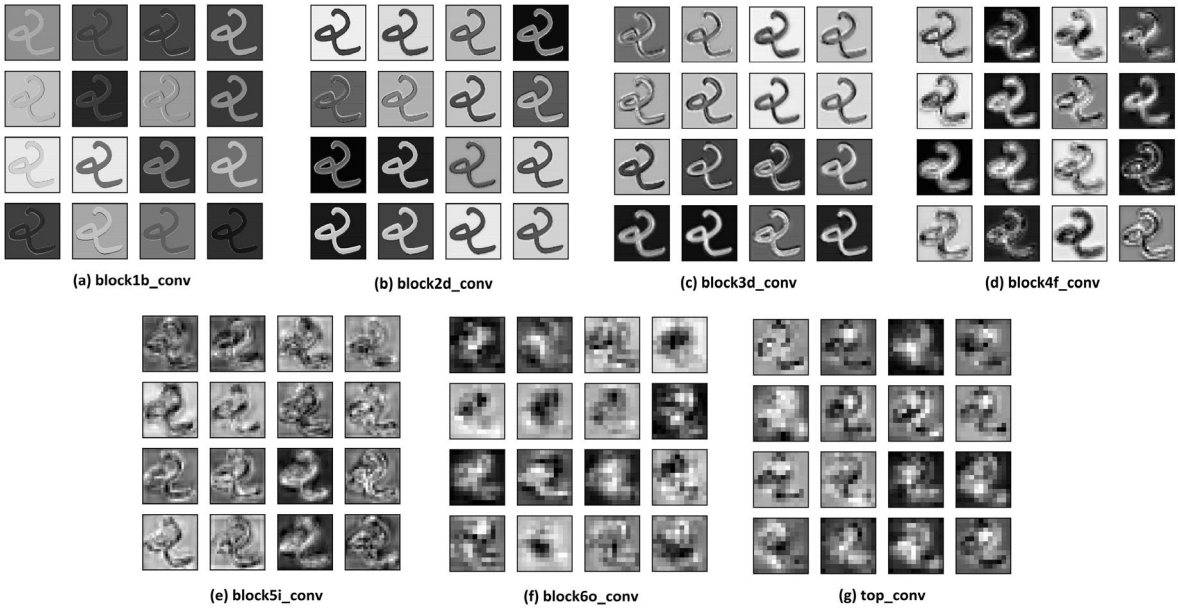

FIGURE 7. Layer-wise weight excitation maps of the highest performing model - EfficientNetV2S.

C. DISCUSSION

In this section, the performance of the three different transfer learning scenarios is compared and also discussed which scenario shows significant performance for handwritten Gujarati numeral classification using deep learning models. We have considered training accuracy, testing accuracy, f1-score, AUC, trainable parameters, and training time for the fair comparison from Tables 1, 2, 3 and 4. In the first scenario, it can be observed that the difference between training and the testing accuracy is more compared to the other two scenarios in Table 1. The first scenario trained the model using an SVM classifier and softmax classifier. Thus, the training time is less than the other scenarios because only the SVM classifier or softmax classifier was trained from the feature vectors. It is also noticed that the testing accuracy, f1-score and AUC are less due to the less number of features transferred in this scenario by freezing the whole network. It is observed that the results of the softmax classifier are better than the SVM classifier from Tables 1 and 2. In the second approach, we can see the promising results of each parameter in Table 3. Because two newly adapted fully connected layers have been added and trained these two layers from the scratch with relu activation function. These layers learned the specific features of our digit dataset well. However, trainable parameters and training time increased in this scenario. In the third approach, it can be seen that model presents average performance compared to the other two approaches in Table 4. To check the feature transferability, the network is chopped in half and the second half of the network was fine-tuned from the scratch hence this approach shows that trainable parameters and training time are the maximum among all the three approaches. However, CPU inference time was reported the same in all three scenarios. Out of all the experiments, the EfficientNetV2S model shows the best performance using the second transfer learning scenario. It can also be observed that the LeNet model shows a marginal difference in testing accuracy and a substantial difference in trainable parameters, training time and model size compared to EfficientNetV2S with the same approach. The layer-wise weight excitation maps of the best performing model are visualized in figure 7 to show how different layers contribute to the overall classification. In figure 7, (a) to (g) depicted the feature visualization of the last convolutional layer of each block and the final convolutional layer. It can be seen that (a) to (c) feature maps learn about the background and foreground in an image. The (d) and (e) feature maps focus on the textures and shapes of a digit. The (f) and (g) feature maps learn specific features of a digit two.

Figure 8 presents the confusion matrix of the EfficientNetV2S model of the second scenario which outperformed in testing accuracy among all the models. The diagonal of this confusion matrix represents correctly classified digits and other numbers except for the diagonal presents misclassified digits. Some samples of misclassified digits are illustrated in figure 9. It is observed that the actual labels of the misclassified images are very close to the shape of the predicted labels. The vertical axis and the horizontal axis of the confusion matrix indicate the true labels and the predicted labels respectively. The shades of blue color show the prediction results and the deeper the blue color, the more accurately model predicated labels of each class.

In addition, the training accuracy, testing accuracy, training loss, and testing loss are depicted in figure 10 – (a) and (b) to know how the model convergence with the number of epochs and achieved the highest accuracy. It can be observed that

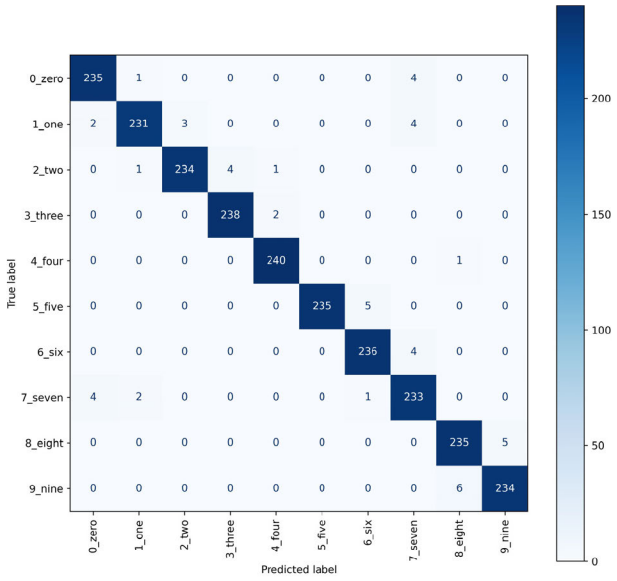

FIGURE 8. Confusion matrix of EfficientNetV2S model using 2nd strategy of transfer learning which showed the best performance among all experiments.

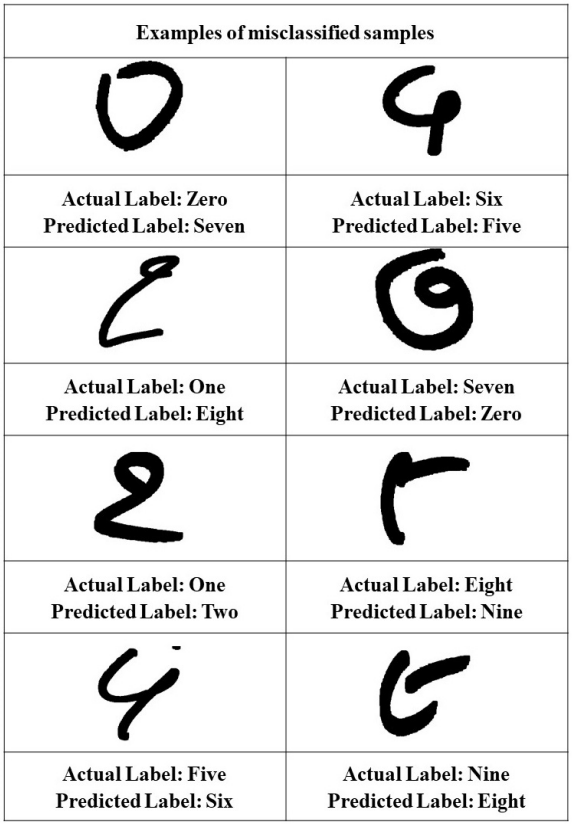

FIGURE 9. Images of misclassified samples.

classification accuracy improves gradually in epochs 1 to 8, then becomes slowly from 9 to 15. Similarly, classification loss decreases rapidly in the initial epochs 1 to 6, then slowly

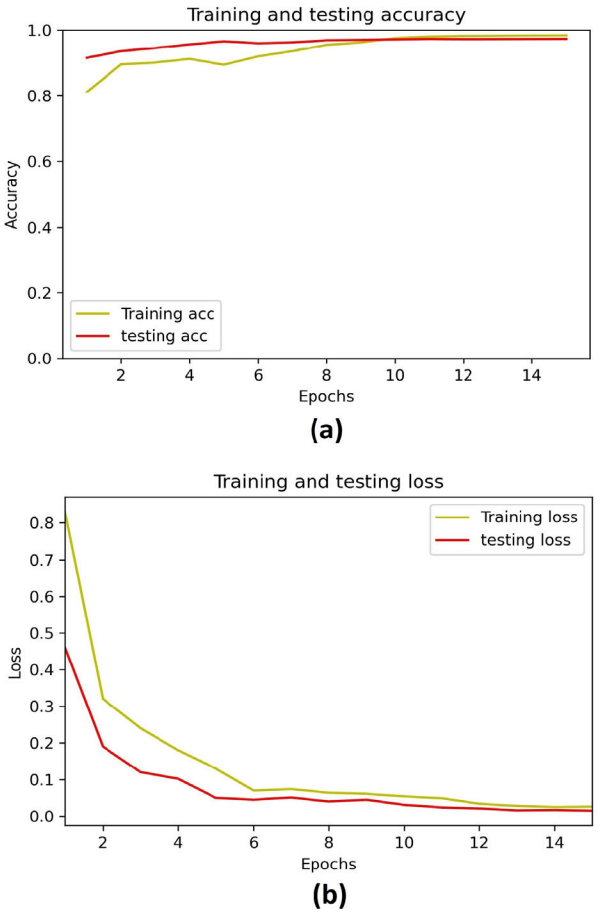

FIGURE 10. Comparison of accuracy and loss during the training of EfficientNetV2S model using 2nd approach of transfer learning; (a) training vs testing accuracy for 15 epochs (b) training vs testing loss for 15 epochs.

TABLE 5. Comparison of our work with SOTA models.

Sr. No.	Authors	Model	Accuracy
1	Limbachiya *et al.* [1]	MobileNet	97%
2	Goel *et al.* [39]	EfficientNetB4	96.5%
3	Proposed Method (Scenario - 1)	EfficientNetV2S	93.75%
4	Proposed Method (Scenario - 2)	EfficientNetV2S	**97.91%**
5	Proposed Method (Scenario - 3)	EfficientNetV2S	96.67%

decreases till the 12th epoch and becomes steady after the 12th epoch. The training and testing accuracies are 98.39% and 97.92% respectively. Furthermore, it can also be seen that training loss and testing loss are near to zero from epoch 12. It indicates that the EfficientNetV2S model provides a better Gujarati handwritten digit classification without overfitting issues or bias and variance error.

Table 5 shows the comparison of our work with other work which are deep learning based approaches mentioned in the literature for the Gujarati handwritten digit classification.

It can be observed that our model using the second scenario outperforms than other models. The accuracy of the other authors' work has been reported from their original research paper.

VI. CONCLUSION AND FUTURE WORK

In this paper, we utilized the deep transfer learning method using ten pre-trained networks for handwritten Gujarati digit classification. We addressed the main issue of transfer learning that how deep to fine-tune the pre-trained convolutional neural network while training the target model. To overcome this issue, we applied three transfer learning strategies to show the effect of the different fine-tuning strategies on the performance of the model. Firstly, pre-trained models were used as feature extractors with a linear SVM and softmax classifiers and it is observed that pre-trained models showed the least performance in this strategy because weight parameters were not fine-tuned during the training on the target task. In the second approach, the pre-trained models were fine-tuned after the last convolutional layer with two newly adapted fully connected layers. We achieved the best performance using this approach because the abstract high-level features were learned by retraining the last few layers. In the third strategy, we noticed the average performance of the models by fine-tuning the models from half of the network and also took maximum time for the training. Finally, experimental analysis was performed using various performance evaluation measures on the self-created handwritten Gujarati digit dataset. We obtained the highest accuracy with the EfficientNetV2S model using the second strategy without training the model from the scratch. We find that the effective fine-tuning strategy of transfer learning improves the performance of the model and significantly reduces the training time. Moreover, it can also conclude that if accuracy is the main measure to decide the top performer, EfficientNetV2S can be the best model and if memory is the main concern to decide a lightweight model for resource-constrained devices, the LeNet model can be the best choice among all models.

In the future, this work can be applied to other regional characters or numerals recognition. Furthermore, our handwritten Gujarati digit dataset can also be utilized to perform research on multiple OCR work for Gujarati numbers.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] K. Limbachiya, A. Sharma, P. Thakkar, and D. Adhyaru, "Identification of handwritten Gujarati alphanumeric script by integrating transfer learning and convolutional neural networks," *Sādhanā*, vol. 47, no. 2, pp. 1–7, Jun. 2022.

[2] L. M. Seng, B. B. C. Chiang, Z. A. A. Salam, G. Y. Tan, and H. T. Chai, "MNIST handwritten digit recognition with different CNN architectures," *J. Appl. Technol. Innov.*, vol. 5, no. 1, p. 7, 2021.

[3] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, and B. Yoon, "Improved handwritten digit recognition using convolutional neural networks (CNN)," *Sensors*, vol. 20, no. 12, p. 3344, Jun. 2020.

[4] N. Bi, J. Chen, and J. Tan, "The handwritten Chinese character recognition uses convolutional neural networks with the GoogLeNet," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 33, no. 11, Oct. 2019, Art. no. 1940016.

[5] A. Alqudah, A. M. Alqudah, H. Alquran, H. R. Al-Zoubi, M. Al-Qodah, and M. A. Al-Khassaweneh, "Recognition of handwritten Arabic and Hindi numerals using convolutional neural networks," *Appl. Sci.*, vol. 11, no. 4, p. 1573, Feb. 2021.

[6] P. Chakraborty, S. S. Jahanapi, and T. Choudhury, "Bangla handwritten digit recognition," in *Cyber Intelligence and Information Retrieval*. Singapore: Springer, 2022, pp. 149–159.

[7] M. A. Akbar and M. S. Islam, "Deep convolutional neural network for handwritten Bangla and English digit recognition," in *Proc. Int. Conf. Electron., Commun. Inf. Technol. (ICECIT)*, Sep. 2021, pp. 1–4.

[8] M. Z. Alom, P. Sidike, T. M. Taha, and V. K. Asari, "Handwritten Bangla digit recognition using deep learning," 2017, *arXiv:1705.02680*.

[9] A. Saini, S. Daniel, S. Saini, and A. Mittal, "Kannadares-next: A deep residual network for Kannada numeral recognition," in *Machine Learning for Intelligent Multimedia Analytics*. Singapore: Springer, 2021, pp. 63–89.

[10] M. Yadav, D. Mangal, S. Natesan, M. Paprzycki, and M. Ganzha, "Assamese character recognition using convolutional neural networks," in *Proc. 2nd Int. Conf. Artif. Intell., Adv. Appl.* Singapore: Springer, 2022, pp. 851–859.

[11] Z.-H. Zhan, J.-Y. Li, and J. Zhang, "Evolutionary deep learning: A survey," *Neurocomputing*, vol. 483, pp. 42–58, Apr. 2022.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2009.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[17] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[21] M. Tan and Q. Le, "EfficientNetV2: Smaller models and faster training," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10096–10106.

[22] S. Antani and L. Agnihotri, "Gujarati character recognition," in *Proc. 5th Int. Conf. Document Anal. Recognit. (ICDAR)*, 1999, pp. 418–421.

[23] J. Dholakia, A. Negi, and S. R. Mohan, "Zone identification in the printed Gujarati text," in *Proc. 8th Int. Conf. Document Anal. Recognit. (ICDAR)*, 2005, pp. 272–276.

[24] J. Dholakia, A. Yajnik, and A. Negi, "Wavelet feature based confusion character sets for Gujarati script," in *Proc. Int. Conf. Comput. Intell. Multimedia Appl. (ICCIMA)*, vol. 2, Dec. 2007, pp. 366–370.

[25] M. Goswami and S. Mitra, "Structural feature based classification of printed Gujarati characters," in *Proc. Int. Conf. Pattern Recognit. Mach. Intell.* Berlin, Germany: Springer, 2013, pp. 82–87.

[26] E. Hassan, S. Chaudhury, and M. Gopal, "Feature combination for binary pattern classification," *Int. J. Document Anal. Recognit.*, vol. 17, no. 4, pp. 375–392, Dec. 2014.

[27] A. A. Desai, "Gujarati handwritten numeral optical character reorganization through neural network," *Pattern Recognit.*, vol. 43, no. 7, pp. 2582–2589, Jul. 2010.

[28] M. Maloo and K. Kale, "Support vector machine based Gujarati numeral recognition," *Int. J. Comput. Sci. Eng.*, vol. 3, no. 7, pp. 2595–2600, 2011.

[29] B. Mj, K. Kv, and J. Me, "Comparison of classifiers for Gujarati numeral recognition," *Int. J. Mach. Intell.*, vol. 3, no. 3, pp. 160–163, Nov. 2011.

[30] A. K. Sharma, P. Thakkar, D. M. Adhyaru, and T. H. Zaveri, "Gujarati handwritten numeral recognition through fusion of features and machine learning techniques," *Int. J. Comput. Syst. Eng.*, vol. 3, nos. 1–2, pp. 35–47, 2017.
[31] V. A. Naik and A. A. Desai, "Online handwritten Gujarati numeral recognition using support vector machine," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 9, pp. 416–421, Sep. 2018.
[32] J. Bharvad, D. Garg, and S. Ribadiya, "A roadmap on handwritten Gujarati digit recognition using machine learning," in *Proc. 6th Int. Conf. for Converg. Technol. (I2CT)*, Apr. 2021, pp. 1–4.
[33] M. Shopon, N. Mohammed, and M. A. Abedin, "Bangla handwritten digit recognition using autoencoder and deep convolutional neural network," in *Proc. Int. Workshop Comput. Intell. (IWCI)*, Dec. 2016, pp. 64–68.
[34] R. Pramanik, P. Dansena, and S. Bag, "A study on the effect of CNN-based transfer learning on handwritten Indic and mixed numeral recognition," in *Proc. Workshop Document Anal. Recognit.* Singapore: Springer, 2018, pp. 41–51.
[35] H. Zunair, N. Mohammed, and S. Momen, "Unconventional wisdom: A new transfer learning approach applied to Bengali numeral classification," in *Proc. Int. Conf. Bangla Speech Lang. Process. (ICBSLP)*, Sep. 2018, pp. 1–6.
[36] A. B. M. A. Rahman, M. B. Hasan, S. Ahmed, T. Ahmed, M. H. Ashmafee, M. R. Kabir, and M. H. Kabir, "Two decades of Bengali handwritten digit recognition: A survey," *IEEE Access*, vol. 10, pp. 92597–92632, 2022.
[37] D. Shukla and A. Desai, "Extraction and recognition of handwritten Gujarati characters and numerals from images using deep learning," in *Proc. Int. e-Conf. Intell. Syst. Signal Process.* Singapore: Springer, 2022, pp. 657–669.
[38] P. Dutta and N. B. Muppalaneni, "DigiNet: Prediction of assamese handwritten digits using convolutional neural network," *Concurrency Comput., Pract. Exp.*, vol. 33, no. 24, Dec. 2021, Art. no. e6451.
[39] P. Goel and A. Ganatra, "A pre-trained CNN based framework for handwritten Gujarati digit classification using transfer learning approach," in *Proc. 4th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Jan. 2022, pp. 1655–1658.
[40] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, pp. 1–40, 2016.
[41] R. Kohavi and F. Provost, "Glossary of terms," *Mach. Learn.*, vol. 30, pp. 271–274, Jun. 1998.

PARTH GOEL received the B.E. (C.E.) degree form Dharmsinh Desai University, Nadiad, Gujarat, in 2011, and the M.E. (CSE) degree from Gujarat Technological University, Ahmedabad, Gujarat, in 2014. He is currently pursuing the Ph.D degree in deep learning with CHARUSAT. He works as the Head of the Department of Computer Science and Engineering, Devang Patel Institute of Advance Technology and Research (DEPSTAR), CHARUSAT. He has ten years of teaching experience. He has guided more than 25 projects at the graduate level. He has published and presented 14 papers in the international conferences, published eight artciles in reputed international journals, wrote five book chapters, and two CSI-magazine articles. He has more than 185 Google Scholar citations. His research interests include deep learning, computer vision, and data science. He has served as a program committee member for international conferences.

AMIT GANATRA (Senior Member, IEEE) received the B.Tech. degree from the Department of Computer Engineering, Gujarat University, Gujarat, India, in 2000, the M.Tech. degree from the Department of Computer Engineering, Dharmsinh Desai University (DDU), Gujarat, in 2004, and the Ph.D. degree in information fusion techniques in data mining from KSV University, Gandhinagar, Gujarat, in 2012. He is currently a Provost with Parul University. He has more than 23 years of teaching experience at bachelor's level and more than 19 years of teaching and research experience at master's level. He is having good interests in teaching, research, and administration. In addition, he is having good research record and published and contributed over 130 articles (as the author and the coauthor) which were/are published in refereed journals and presented in various national and international conferences. He has supervised (guided) more than 100 industry projects at graduate level and more than 100 dissertations at postgraduate level and 12 Ph.D scholars have completed their Ph.D. under his supervision and he is also supervising (guiding) eight research scholars at Ph.D. level. He has more than 2300 Google Scholar citations. His research interests include database technologies, data warehousing and data mining, data analytics, data science and big data analytics, artificial intelligence, machine learning and deep learning, soft computing, object oriented concepts, cloud computing, system software, the IoT, and software engineering. He is an active member of ACM and CSI Professional Society Chapters. He has served as the program committee chair for many international conferences. He is a reviewer of Scopus and Web of Science indexed journals.