Soft Computing
https://doi.org/10.1007/s00500-020-05368-8

METHODOLOGIES AND APPLICATION

Arabic (Indian) digit handwritten recognition using recurrent transfer deep architecture

Rami S. Alkhawaldeh[1]

Abstract
The rapid volume of digit texts and images motivates researchers to build solid and efficient prediction models to recognize such media. The Arabic language is considered one of the difficult languages regarding the way of writing characters and digits. Recent research focuses on such language for building predictive approaches to recognize written materials. The Arabic (Indian) digit recognition task has been a challenging task and has gained more attention from researchers who build optimal predictive models from historical images that are used in many applications. However, transfer learning approaches exploit deep pre-trained models that could be re-used for similar tasks. So, in this paper, we propose an adapted deep hybrid transfer model developed using two well-known pre-trained convolutional neural networks (CNN) models. These are further adapted by adding recurrent neural networks especially long short-term memory (LSTM) architectures to detect Arabic (Indian) Handwritten Digits (AHD). The CNN model learns the relevant features of Arabic (Indian) digits, while the sequence learning process in the LSTM layers extracts long-term dependence features. The experimental results, using popular datasets, show significant performance obtained by the adapted transfer models with accuracy reached up to 98.92% as well as with precision and recall values at most cases reached to 100% with statistical t test using p-value ($p \leq 0.05$) compared to baseline methods.

Keywords Arabic (Indian) handwritten recognition · Deep supervised learning · LSTM · Transfer learning

1 Introduction

Handwritten digit recognition (HDR) is being one of the attractive areas in the fields of image processing and pattern recognition (Abdleazeem and El-Sherif 2008; AlKhateeb and Alseid 2014; Loey et al. 2017; Mudhsh and Almodfer 2017; Dargan et al. 2019; Elleuch et al. 2016). The crucial importance of HDR is in deploying effective techniques to detect the written digits in a more accurate and speedy manner. HDR is also used as a benchmark for comparing different classification techniques. Specifically, the optical character recognition (ORC) is a sub-task of HDR problem that includes some applications such as postal code and bank cheque reading, automatic pin code recognition and collecting data from filling in forms (Maheshwari et al. 2018). The original works of OCR handwritten alphabets and numerals started with Latin languages. Several works show an efficient performance related to these languages using printed characters rather than handwritten ones. Although these works achieved better performance, other languages such as Arabic are in the exploration and analysis phases. This is required to build effective predicted models to reach high predicting performance. The reason what characterizes the Arabic language from other languages is the way of typing words and characters from right to left, while the numeric digits are written from left to right. Younis (2017) argues that Arabic HDR suffers from slow development compared to handwriting recognition in other languages. Thus, recognizing Arabic (Indian) handwritten digits is a challenging task and suffers from several problems such as different writing style, shapes, and some written images including noise (Mahmoud 2008). Figure 1 shows the original digits that are written correctly and some digits that are written badly. As shown in Fig. 1b, number 2 is badly written, number 7 is approximately similar to number 6, number 3 is similar in structure to number 2, number 0 is written like number 1, the style of number 4

Communicated by V. Loia.

✉ Rami S. Alkhawaldeh
r.alkhawaldeh@ju.edu.jo

[1] Department of Computer Information Systems, The University of Jordan, Aqaba 77110, Jordan

Published online: 09 October 2020

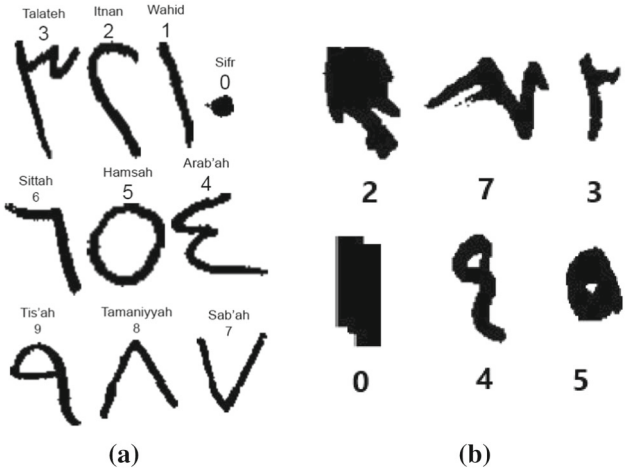

Fig. 1 The original and corrupted digits (Mahmoud 2008) **a** Arabic (Indian) digits and **b** badly written digits

is similar to number 9, and number 5 is close and confuses in shape to number 0. Therefore, based on such reasons, an effective predictive model is indeed required to recognize the digits for applications written in the Arabic language.

Several research efforts have risen to develop a predictive model using Machine Learning (ML) techniques to recognize digits of various writing forms (Selvi and Meyyappan 2013; Mudhsh and Almodfer 2017; Younis 2017). Deep learning has emerged as an outstanding field in artificial intelligence applications such as computer vision, natural language processing and speech recognition with effective results (LeCun et al. 2015; Kim et al. 2020; Alkhawaldeh 2019; Khawaldeh et al. 2018; Alkhawaldeh et al. 2019). It requires a huge amount of raw data to extract features as inputs for classifier models. In deep learning, an approach called transfer learning is being used. That is, a machine learning method exploits a pre-trained model or patterns on a task and re-uses it for other identical tasks. A pre-trained model is trained on a large benchmark dataset of a specific problem and later used to solve similar problems. The reason for exploiting the pre-trained models as popular approaches in deep learning is because of the similar availability of current applications. This reduces the computation and time-consuming in the training process phase (Canziani et al. 2016; Rawat and Wang 2017). Hence, several state-of-the-art results in image classification depend on transfer learning solutions (Krizhevsky et al. 2012; Simonyan and Zisserman 2014; He et al. 2016; Pan and Yang 2009).

The pre-trained models, used in transfer learning, include large CNN models trained on well-known datasets (Voulodimos et al. 2018; Bengio 2009). A CNN model has two separates: the convolutional base and classifier parts, as shown in Figs. 2 and 3. The Convolutional base comprises a stack of convolutional and pooling layers that aims to generate the most significant features from the input images (Chollet 2018). In sequences of operations, a set of filter maps or kernels, represent the weights of the network, are initialized to slide over the image for deriving relevant features in a form of updated weights. These feature maps are then sub-sampled using pooling layers to reduce their sizes, which at the same time reduces the size of the input images. Using such a process in separate stacks leads to extracting features from low-dimensional pixel images- with common features re-used in various problem domains at a few first layers- to more combined images of different features at last layers of the CNN model. In the classifier part, fully connected artificial neural network (ANN) layers are used to classify the images using the detected components gathered from the Convolutional part. Many researchers have conducted experiments to examine various ML techniques such as support vector machines (SVM) instead of fully connected ANNs.

In transfer learning, there are three strategies used to train the architecture models (Goodfellow et al. 2016; Alom et al. 2019): which are (i) training the entire proposed model which includes CNN and classification parts, (ii) train some layers and leave the others frozen and (iii) freeze entirely the convolutional base layers. There are a set of architecture models proposed as classifiers for image datasets such as VGG (Simonyan and Zisserman 2014), InceptionV3 (Szegedy et al. 2016), and ResNet5 (He et al. 2016). The skeleton of these models, in the first scenario, is utilized where the weights are revised from scratch on convolutional base layers to the classification part. In the second strategy, a set of layers from the convolutional and classifier layers are selected where their weights are frozen during the training process of the model. The users have to choose how many layers are not to be updated during the training phase to adjust the weights of the network. With freezing the convolutional base layers, the convolutional base layers are handled to be utilized as feature extraction where their weights are suspended from the updating process to be inputs for the classifier part. This is useful in reducing training power and time-consuming. Here, the architecture of the pre-trained model is used and trained according to the dataset.

The issue is how to use transfer learning to recognize the digits in applications of writing digits (Indian) in the Arabic language. In this study, we exploit two transfer learning models to evaluate their performance in predicting these digits (Indian) in Arabic expression. Our contribution maintains updating two common deep learning architectures that are GoogleNet and VGG-16. The updates include joining LSTM (long short-term memory network) (Hochreiter and Schmidhuber 1997) layers to the network architectures before the fully-connected dense layers in the network. The convolutional and pooling layers of the two transfer learning models are promoted with the LSTM layers in the classification part connected to the dense layers. This joining process of the

LSTM layers to the architectures learns more critical features from the former layers.

To our knowledge, this is the first time such a model is used for Arabic (Indian) digit recognition using three CNN, LSTM, and fully-connected layers on large datasets. The question is then does the hybrid transfer model mitigate the challenge of recognizing the Arabic written (Indian) digits that suffer from their writing style and noise. Furthermore, we improve the performance of the hybrid transfer model by using several optimizing techniques such as regularization and data augmentation. These techniques handle the problem of overfitting to generalize the model to future test sets.

The structure of the paper is organized as follows: Sect. 2 addressed a set of related works on Arabic (Indian) handwritten recognition using traditional and deep learning techniques. The section also presents the accuracy of the studied techniques. Section 3 presents and discusses our adapted transfer learning techniques and their building block architectures for recognizing the Arabic (Indian) digits. Section 4 shows the experimental settings such as the datasets and the empirical results. Our discussion and conclusions along with future work are summarized in Sect. 5.

2 Related work

In recent decades, several research studies have emerged for the Arabic (Indian) HDR. These are categorized into two types: classical classification and deep learning techniques. This paper presents proposed approaches for Arabic (Indian) HDR, hence a large dataset is required for evaluating the models. The most popular dataset for handwritten digits is MNIST proposed by LeCun et al. (1998). To build similar dataset on Latin and Arabic (Indian) digits, researchers developed a dataset to allow for a more direct comparison of the performance of classification algorithms. The largest dataset for the Arabic (Indian) HDR is proposed by Abdleazeem and El-Sherif (2008) dubbed MADbase. This causes an efficient training process for the network and fine-tuning parameters. The MADbase is a modified version of the dataset called ADbase with same images and different size. To constructing MADbase to be comparable to MNIST, ADbase images were resized and transformed from binary to grey-scale. In this work, we use such a dataset because of its size for comparison and evaluation purposes.

In classical classification techniques, Mahmoud (2008) proposed a method to recognize handwritten Arabic (Indian) digits using a dataset of 21,120 images provided by 44 writers. This approach depends on extracting Gabor-based features and support vector machines (SVMs). The results showed that the average classification accuracy rates were 99.85% and 97.94% using three scales & five orientations and four scales & six orientations. Abdleazeem and El-Sherif (2008) carried out experiments on several classification methods to the MADbase dataset where radial basis function support vector machine (RBF-SVM) showed the best results among the alternative algorithms. They conducted the classification scheme in a two-stage classification. First, the researchers extracted a set of features from the dataset and then used them as input for the RBF-SVM. The RBF-SVM classifier tuned its parameters to maximize the classification accuracy, reaching 99.48% from the best parameter combination values. Selvi and Meyyappan (2013) proposed neural networks for Arabic (Indian) digit recognition that were built using the back-propagation approach on a small dataset. The classification accuracy was 96%. AlKhateeb and Alseid (2014), using the ADbase dataset and by using Dynamic Bayesian Networks (DBN), got an accuracy result of 85.26% classification. Salameh (2014) suggested two methods for improving Arabic (Indian) Handwritten Digits recognition. These methods combine fuzzy logic pattern classification to estimating the number of ends related to the digit shapes. The results revealed a classification accuracy of 95%.

Because of the effective results of deep learning techniques, a set of approaches has recently been proposed. Their results showed high performance compared to traditional classification techniques. Loey et al. (2017) used Stacked Autoencoders on the MADbase dataset result in a classification accuracy of 98.5%. By using dropout regularization, data augmentation and an inspired VGGNet model, Mudhsh and Almodfer (2017) obtained a validation accuracy of up to 99.66% on the MADbase dataset. The validation method was the tenfold cross-validation technique. In addition, they tested the algorithm performance on a dataset of 6,600 images of characters with validation accuracy of 97.32%. Younis (2017) obtained an accuracy of 97.60% on AHCD dataset, by using a Deep CNN with batch normalization and learning rate parameter. The conducted experiments showed that the CNN models obtained high-performance results compared to conventional ML techniques.

3 Recurrent deep transfer architectures

We consider the Arabic language as one of the difficult languages in the style of writing characters by hand. Therefore, it is substantial to find optimal or sufficient solutions for handwriting recognition. Deep transfer learning systems are promising solutions that have shown significant results in recent research. These techniques need huge data, as it has a low performance on a few data. The proposed architectures, in this work, adapt deep transfer models that follow a set of phases to build an effective, predictive, or optimal classifier for Arabic (Indian) digit handwritten recognition. These phases include feature extraction and classification parts. In the learning features phase, the proposed adaptive models

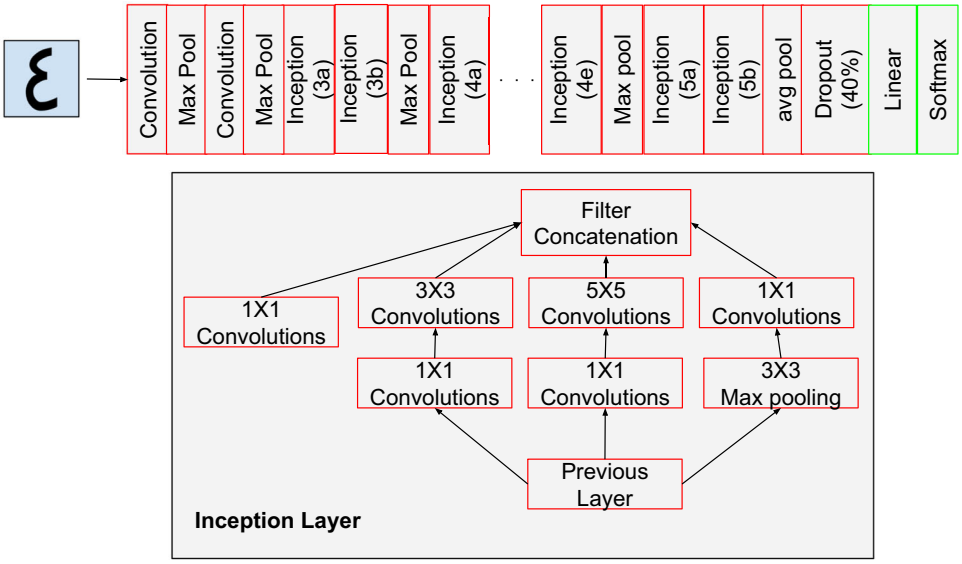

Fig. 2 LeNet deep learning architecture

Fig. 3 VGG-16 deep learning architecture

receive a set of Arabic (Indian) digit images, each of which passes in sequences of convolution and pooling operations. The learning operation, during such sequences, extracts the most relevant features from high-level abstraction to a lower dimensional phenomenon. In the classification phase, we exploit the extracted features to be as inputs to any selected ML techniques to map the input features to output targets (i.e. ten digits). However, the adaptive deep transfer architectures exploit the trained weights of two models to connect them with RNN models and later with fully connected ANN. The two transfer learning models for Arabic (Indian) digits are GoogLeNet (Inception V1 or LeNet) and VGG-16 models. In a nutshell, the hybrid transfer model uses CNN layers as feature extraction part then connects these features with recurrent LSTM layers for the learning sequence features and the final relevant features are considered as input for fully-connected ANNs.

3.1 Transfer deep model for features learning and extraction

The essential core of the adaptive architectures are the two-deep transfer models and the connected recurrent LSTM model. These architectures show significant performance in predicting digits; those are written by hand. The two deep transfer architectures GoogLeNet and VGG-16 models are shown in Figs. 2 and 3 where the input is number 4 in Arabic (Indian).

The GoogLeNet model comprises 27 layers including two normal convolutions, 9 inception layers (or modules), 4 max & 1 average pooling layer and 40% drop out in the CNN part. In addition, it uses linear dense layers and Softmax layer in the classification part. Each part of the model involves a crucial role in the network. The convolution maintains the image pixel relationship by using small squares of input data to learn image features. Within convolution, the inception modules are a set of convolution layers that are connected as a branching manner for expanding the network with more deep layers. The inception layer comprises a set of components including convolutions with different size of two processes to aggregate and concatenate the filter features as a parallel operation. The idea of inception layer is back to that neurons fire together are wired together as a compact layer. In specific, the inception layer comprises 1X1 convolutions, 3X3 convolutions, 5X5 convolutions, and 3X3 max pooling that are connected from the previous input, and stack together at output filter concatenation. The 1X1 convolution is used in the inception module for dimension reduction as it like pooling operations. The max & average pooling layers reduce the size of filters to update a few weights that in which increases the computational power and drops the time complexity. The dropout operation drops a set of neurons or connection weights to achieve the best fit model and deal with the problem of over-fitting (Rosa et al. 2018). On the other hand, the VGG-16 model comprises 19 layers with-

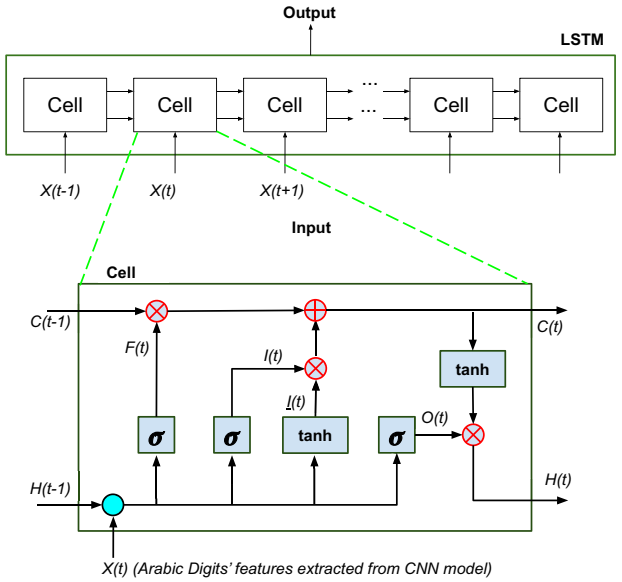

Fig. 4 Building Block of the LSTM Architecture

out branching as in the inception layer and uses two pooling methods; max & average pooling layers.

3.2 Recurrent LSTM layers

What characterizes the adapted approaches comparing to the original models is that it uses the LSTM as a base layer between the CNN model and fully connected dense layers of the classification parts on both models. The LSTM layers represent a sequence of layers that are iterative to persist the feature maps obtained from the output of the CNN model as input sequence features. In more detail, the LSTM models are a special kind of the Recurrent Neural Network (RNN) models which have a chain repeating modules form of neural networks. In the RNN models, the connections of forwarding information in the network depend on the time steps; which means that the output at a specific time step becomes the input to the next time step. Specifically, at each element of the input sequence, the model also considers what it is persisting from the preceding elements. This phenomenon avoids the long-term dependency problem. Therefore, the architecture of the LSTM models has a complex recurrent structure expressed in a single cell that is connected in time, where the most popular cell is the long short-term memory, as shown in Fig. 4. This maintains a cell state and a carry for ensuring that the information in the form of a gradient is not wasted as the sequence is processed.

As shown in Fig. 4, the LSTM comprises two critical values: the first one is the hidden state $H(t)$ value of the cell that is updated by the time, and the other one is the cell state $C(t)$ that handles memory in the long term. The cell state is changed along the top line of the LSTM cell. The LSTM can add or remove information in the cell state where the function of each cell element is determined by the parameters (weights) learned during training. The cell elements of the LSTM models are forget, input, and output gates. The forget gate $F(t)$ adjusts the connection of the input $X(t)$ and the previous hidden state $H(t-1)$ to the cell state $C(t)$. This allows the cell to remember or forget $X(t)$ where the $H(t-1)$ based on the binary output of the sigmoid functions. The input gate $I(t)$ and $\underline{I}(t)$ determines whether to feed the input value to the cell state $C(t)$. The output gate is used for producing predictions at each time step. The output gate $O(t)$ determines the final value based on the cell state $C(t)$. This process is formulated in a set of equations shown in Eqs. 1 – 6. There are four major functions used in LSTM models: sigmoid (σ), hyperbolic tangent ($tanh$), multiplication ($\times$), and sum ($+$). These functions make the processes easier to update the weights during the back-propagation process.

$$F(t) = \sigma(W_f[H(t-1), X(t)] + B_f) \tag{1}$$
$$I(t) = \sigma(W_i[H(t-1), X(t)] + B_i) \tag{2}$$
$$O(t) = \sigma(W_o[H(t-1), X(t)] + B_o) \tag{3}$$
$$\underline{I}(t) = \tanh(W_i[H(t-1), X(t)] + B_i) \tag{4}$$
$$C(t) = F(t) \cdot C(t-1) + I(t) \cdot \underline{I}(t) \tag{5}$$
$$H(t) = O(t) \cdot \tanh(c(t)) \tag{6}$$

where W and B denote the weight matrix and bias vector, respectively. $\sigma(\cdot)$ stands for the sigmoid function and $\tanh(\cdot)$ denotes the hyperbolic tangent function. W_f, W_i, W_o denote the weights for forgate, input and output gates. The reason for using LSTM layers is that of its robustness to recognize the digits. The LSTM also can process the longest sequence data without appearing the vanishing of the gradient. The vanishing gradient shows a slight decrease in the values of the weights, which appears no changes during the training process. The back-propagation gradient decent updates the weights of whole networks including the LSTM layers' weights.

The reason for using LSTM architecture is that the forget gate is exploited to control the flow of gradient in the network which solve the problem of gradient vanishing problem (Hochreiter 1998). This process also refines the input information by cleaning the noise and redundant information. In terms of images, the process stores and refines the relevant shape of an input image in a particular pattern by filtering the dependencies between the values feature maps extracted from the CNN network. Hence, the effective features are stored in cell state for a long time while the invalid features are forgotten.

As a result, the proposed models are the CNN model as a feature extraction part, LSTM layers for learning sequences in the extracted features. Then the LSTM outputs are passed through fully connected layers in the classification part.

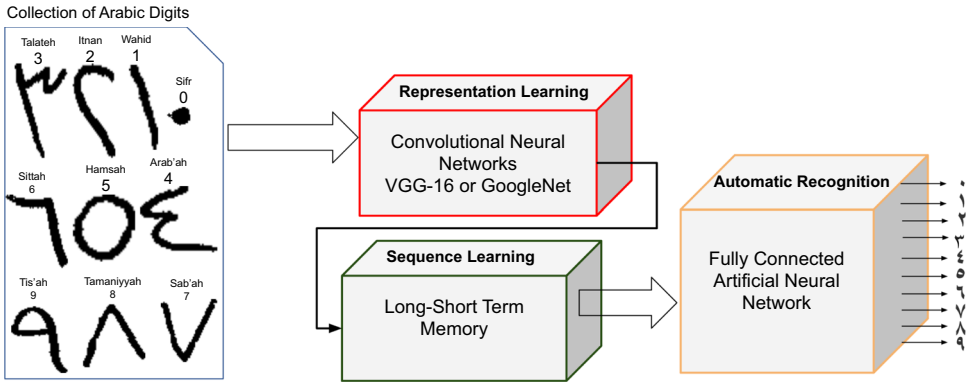

Fig. 5 An illustration of the Arabic (Indian) digits automatic detection system

Thus, we develop the hybrid adaptive CNN+LSTM -based technique involving both the representation and sequence learning stages to detect Arabic (Indian) digits. Figure 5 gives a block diagram of the hybrid adaptive model applied in this study.

4 Experimental environment and results

To evaluate the proposed hybrid adaptive CNN+LSTM -based technique, this section presents the experimental environment and classification results. In the experimental environment, hardware and software specifications, the datasets, parameter settings, evaluation metrics, and a technique, to increase the number of images for enhancing the performance, are distinctly demonstrated. In classification results, the evaluation results and comparisons with the baseline techniques are discussed.

4.1 Experimental environment

To evaluate the proposed adaptive models, the experiments are performed using Intel(R) Core (TM) i7-1065G7 (8 CPUs), NVIDIA GeForce MX230 GPU (12 GB), and 16 GB RAM. The software tools for conducting the experiments are Tensorflow[1] v2.2.0 using Keras deep learning libraries[2] v2.3.1 as back-end.

This work uses two large Arabic datasets which are Arabic (Indian) Digits Handwritten dataBase (AHDBase) and Modified AHDBase (MAHDBase)[3]. The AHDBase is composed of 70,000 digits written by 700 participants. To ensure including different writing styles, the database was gathered from variant institutions. They partition the database into two sets: a training set (60,000 digits – 6000 images per class) and a test set (10,000 digits – 1000 images per class). For a comparison perspective, the two databases of Arabic (Indian) and Latin digits should be of the same format. Because of the value of the MNIST dataset for Latin handwritten digits, they have created the MAHDBase with equal size to the AHDBase and the same size and format of MNIST dataset.

This work considers using the image data augmentation technique used in much deep transfer learning works (Perez and Wang 2017). The image data augmentation is a technique that produces a variant collection of modified images from the training dataset to expand its size. The images that are produced using data augmentation technique involve transformed versions of images in the training dataset belong to the same class as the original image. The transformation process includes a range number of operations derived from the field of image processing, such as shifts, flips, zooms, etc. By increasing the size of the training dataset, this produces a variation of the training dataset images is most likely perceived by deep learning models. This overwhelms the models with a huge number of images. This results in high performance that improves the ability of trained models to generalize what they have learned to new images. In Arabic (Indian) digits image recognition, the method of data augmentation has been used by Mudhsh and Almodfer (2017) with high-performance results.

We address the data augmentation to the training sets of both AHDBase and MAHDbase. The purpose is to create varied images of the dataset that make the classifier more powerful. The data augmentation parameters including the image processing operations and specific ranges are: rotation angles {30°, 90°, 180°, 270°}, the same width and height of shifting, shearing and zooming as {0.1, 0.3, 0.5} and using horizontal flipping. The general hyper-parameter settings of the model are given in Table 1.

For comparison perspective, we use the original GoogleNet and VGG-16 models as baseline methods for evaluating the performance of the proposed models. The evaluation metrics that are used are accuracy, precision and recall. In addition, the validation method is a tenfold cross-validation technique during training for building the final models.

[1] https://www.tensorflow.org/

[2] https://keras.io/

[3] http://datacenter.aucegypt.edu/shazeem/

Table 1 Various parameter settings used to create the adaptive models

Parameters	Values
Optimizer	Adam
Learning rate	1e-4
Epoch	75
Batch size	32
Decay	1e-4 / Epoch
Loss function	Categorical cross entropy

– Accuracy refers to the number of corrected classified instances to the total number of instances.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{7}$$

– Precision refers to the fraction of relevant instances among the retrieved instances.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

– Recall (known also as sensitivity) is the fraction of relevant instances that have been retrieved over the total relevant instances.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

where TP refers to the positive instances that are correctly classified as positives. TN refers to the negative instances that are correctly classified as negative. FP refers to the negative instances that are incorrectly classified as positive. FN refers to the negative instances that are incorrectly classified as positive

4.2 Classification results

This study conducts a set of experiments using Arabic (Indian) digits datasets. The experiments included constructing adapted models using the training set of 6000 image instances and validation set involving a subset of images selected from the training set. Besides the training phase for building the adapted models, we use another test set of 1000 instances of images for evaluation. Both training and test sets involve more varied images that are created by using the data augmentation technique. In training and validation phase, the adapted GoogleNet and VGG-16 models with LSTM layers obtained better performance using 70 epochs on both datasets, as shown in Figs. 6 and 7. These figures explain the loss and accuracy values of the training and validation phases. The loss values, using proposed models in two datasets, decline after every epoch (100 number of iterations) in a steady form from 20% up to 70%. As shown, the clear gaps between values obtained for training and validation loss are close to each other. This means achieving the right fit models to alleviate or ignore the under-fitting and over-fitting problems. The constructed fitted models are still suitable for classifying the future written digits taken into consideration in generalization to new images.

Regarding accuracy results, the adapted GoogleNet and VGG-16 models show high performance in recognizing training and validation sets using tenfold cross-validation technique. They obtain these results in a few epochs, whereas updating weights using the back-propagation technique is stable in the early stages. The training and validation accuracies on both datasets increase from approximately 80% using adapted GoogleNet, while it increases using the adapted VGG-16 model from approximately 85% at 10 epochs. The results show much more matching between training and validation accuracy values from the early stages of training

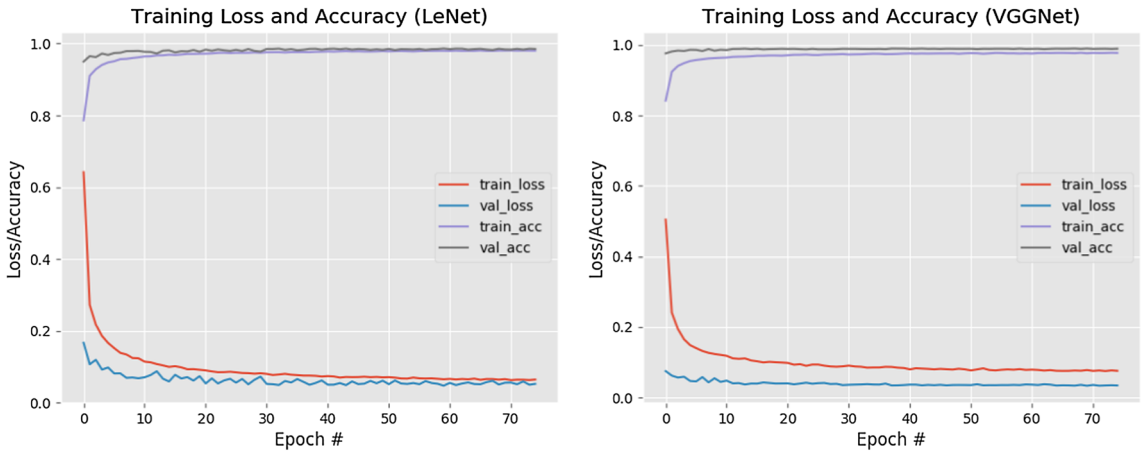

Fig. 6 Training and Validation loss using AHDBase dataset **a** LeNet+LSTM and **b** VGG-16+LSTM

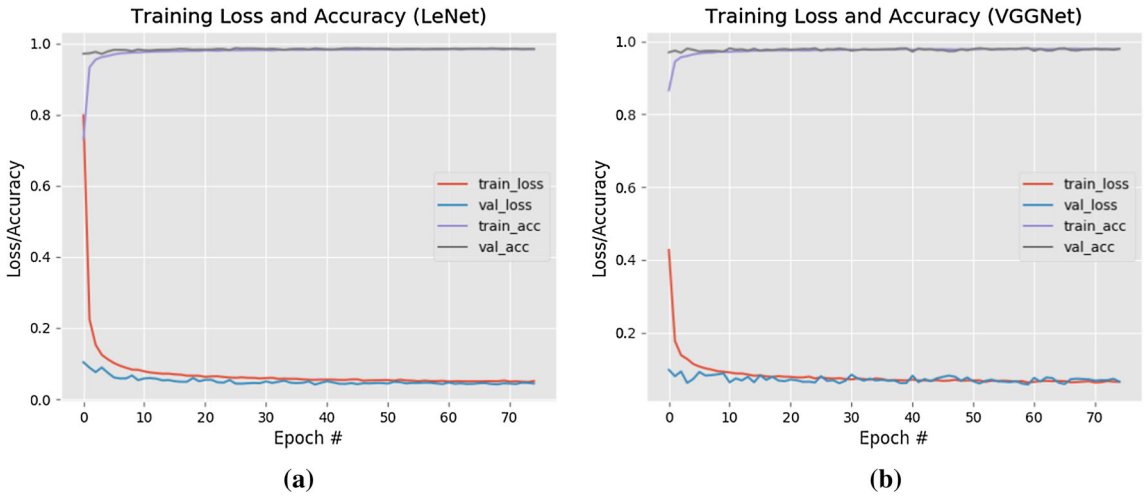

Fig. 7 Training and Validation loss using MAHDBase dataset **a** LeNet+LSTM and **b** VGG-16+LSTM

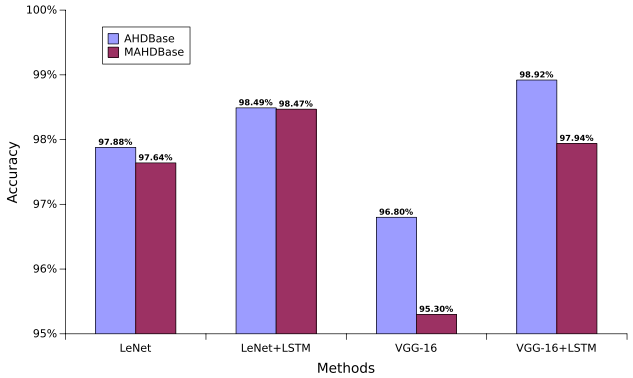

Fig. 8 The Accuracy comparison of the Adapted models and Baseline methods

models; which shows the robustness of models regarding predicting digits. The proposed models obtain a better performance to a classifier for Arabic (Indian) digits, because of the speed in convergence with fitted models at a decreasing number of epochs and high accuracy values in predicting digits.

The experiments in the training phase created highly effective classifier models to be used for recognizing future unknown instances. The accuracy of these models on the test set is shown in Fig. 8. The x-axis illustrates the proposed and baseline methods, while the y-axis represents the accuracy values. The accuracy of models that are conducted on the AHDBase dataset obtains better performance compared to the result using the MAHDBase dataset on the adapted and baseline methods. This reduction in values backs to images' shapes and sizes in MAHDbase dataset. The VGG-16+LSTM model shows better accuracy value of 98.92% on AHDBase dataset compared to LeNet+LSTM model of 98.49%. The LeNet+LSTM model, using MAHDBase dataset, obtains high accuracy of 98.47% which is more than the VGG-16+LSTM of 97.94%. In comparison to baseline methods, the LeNet+LSTM and VGG-16+LSTM models outperform the original baselines on both datasets with significant improvements using VGG-16+LSTM model. In summary, the results have shown the importance of using the adapted models as classifiers for future unknown images. This leads to choosing these models as promising classifiers that could be used for Arabic (Indian) digits or characters, especially those written by hand.

The accuracy value sometimes does not show a real evaluation of the constructed models. This is because of the typical nature of the accuracy metric that estimates the corrected instances over all the instances of a test set; in which the focus is on the data. Therefore, we use two essential metrics for evaluation that show deep results based on correct and retrieved instances: precision and recall. The precision metric represents the percentage of correctly predicted positive (or real) digit values by the model to the total predicted positive values. The recall (or sensitivity) metric is the ratio of correctly predicted positive values by the model to the actual positive values.

Tables 2 and 3 show the results of the adapted and baseline models using precision and recall metrics along with two-paired statistically significant t-test of $p \leq 0.05$ denoted as *. The first column represents the 10 digits as written in Fig. 5, while the last row shows the average of the two metrics on the adapted and baseline models including the standard deviation values.

In AHDBase dataset, the adapted transfer models obtain high precision and recall values reached approximately on average to 99% ($\pm$ 0.009) precision and 99% ($\pm$ 0.004) recall for LeNet+LSTM model and 98% ($\pm$ 0.011) recall and 99% ($\pm$ 0.005) precision for VGG-16+LSTM model. The same is for the results using MAHDbase that obtain significant performance using the adapted transfer models on

Table 2 Arabic (Indian) Numerical Transfer learning using AHDBase: two-paired statistically significant t-test; $p \leq 0.05$ are denoted as *

Number	Precision				Recall			
	LeNet	LeNet +LSTM	VGG-16	VGG-16 +LSTM	LeNet	LeNet +LSTM	VGG-16	VGG-16 +LSTM
0	0.98	**1**	0.97	**0.98**	0.9	**0.94***	0.98	**0.99**
1	0.91	**0.94***	0.97	**0.99***	0.97	**0.99**	0.99	0.99
2	0.97	0.97	0.95	**0.97**	0.97	**0.98**	0.94	**0.98***
3	0.96	**0.99***	0.97	**0.99**	0.98	**0.99**	0.98	0.98
4	0.98	**0.99**	0.98	**0.99**	0.97	**0.99**	0.97	**0.99**
5	0.98	**0.99**	0.99	**1**	0.98	**0.99**	0.98	**0.99**
6	0.97	**0.99**	0.96	**0.98***	0.93	**0.98***	0.96	**0.99***
7	0.98	**1***	0.99	**1**	0.98	**1***	0.99	**1**
8	0.98	**0.99**	1	1	0.98	**0.99**	0.99	0.99
9	0.97	**0.99**	0.99	0.99	0.96	**0.99***	0.98	**0.99**
Average	0.97	**0.99**	0.97	**0.99**	0.98	**0.99**	0.99	**0.99**
(stdev)	(±0.02)	(±0.009)	(±0.016)	(±0.011)	(±0.027)	(±0.004)	(±0.02)	(±0.005)

The bold values illustrate the improvement of Recall or Precision values on the new technique in comparison with baseline method like for example LeNet+LSTM and baseline method LeNet

Table 3 Arabic (Indian) Numerical Transfer learning using MAHDBase: two-paired statistically significant t-test; $p \leq 0.05$ are denoted as *

Number	Precision				Recall			
	LeNet	LeNet +LSTM	VGG-16	VGG-16 +LSTM	LeNet	LeNet +LSTM	VGG-16	VGG-16 +LSTM
0	0.98	0.98	0.99	**1**	0.98	0.98	0.865	**0.93***
1	0.98	**0.99**	0.95	**0.97**	0.97	**0.99**	0.97	**1***
2	0.97	**0.98**	0.87	**0.91***	0.95	**0.97***	0.97	**0.99**
3	0.99	0.99	0.99	0.99	0.99	0.99	0.98	**0.99**
4	0.99	0.99	0.99	**1**	0.99	**1**	0.97	**0.98**
5	0.98	**0.99**	0.94	**0.97***	0.97	**0.98**	0.99	0.99
6	0.91	**0.95***	0.96	**0.98***	0.93	**0.97***	0.89	**0.94***
7	0.97	**1***	0.99	**1**	0.99	**1**	0.98	**1**
8	0.99	0.99	0.99	0.99	0.99	0.99	0.98	**0.99**
9	0.96	**0.99***	0.98	**0.99**	0.97	**0.99**	0.97	**0.99**
Average	0.97	**0.99**	0.97	**0.99**	0.97	**0.99**	0.96	**0.99**
(stdev)	(±0.024)	(±0.005)	(±0.038)	**(±0.011)**	(±0.020)	**(±0.008)**	(±0.043)	**(±0.006)**

The bold values illustrate the improvement of Recall or Precision values on the new technique in comparison with baseline method like for example LeNet+LSTM and baseline method LeNet

average precision 99% (± 0.005) and 99% (± 0.011) for LeNet+LSTM and VGG-16+LSTM respectively. They have high recall values of 99% (± 0.008) and 99% (± 0.006) for LeNet+LSTM and VGG-16+LSTM respectively. Furthermore, the adapted models provide significant results in comparison to baseline models at specific characters on both datasets. For example, number 7 has 100% significant precision and recall values using adapted models. These results indicate that using LSTM layers as a part of the baseline models is a potential choice for future experiments.

5 Conclusion and future work

The transfer deep learning architectures convey the learned knowledge from a specific model to other models of similar applications. However, because of the importance of detecting the written Arabic (Indian) digits and texts, this motivates to build efficient deep learning models to recognize such information. In this work, we used two well-known transfer models for Arabic (Indian) Digits Recognition with some modifications as adapted models: the GoogleNet+LSTM and VGG-16+LSTM models.

The idea is to include the LSTM layers as a part in these models to persist the preceding extracted features from the CNN part. The adapted transfer models comprise three parts: the CNN model, the LSTM layers, and the fully connected dense layers. This architecture block increases the strength of the model for recognising the input digits by using the LSTM layers as a kind of RNN models. The LSTM layers, by using cells, ignore irrelevant information and keep the most important ones. Then, the final output of the LSTM layers is the most relevant information that has to be inputs for the fully dense layers. Hence, the main core of this study is the advantage of its high performance in detecting Arabic (Indian) digits by using both local characteristics presented by feature maps and long-term dependencies of the input images. The CNN networks learn local properties while the LSTM network learns sequences on these features.

The results showed how efficient is to deploy LSTM layers as the main part for learning long-term dependencies using deep transfer models. During the training phase, the experiments increase the performance result in decreasing the loss values at 25 epochs with accuracy values raised approximately from 92%. This means that the computation complexity of the building process could be determined using a few epochs to bring in the early stage high model performance. However, each digit generates a different outcome in precision and recall because of using the data augmentation technique. This technique uses transformation methods to create variant images from the original dataset by some operations such as image rotation. These operations generate transformed images that influence the experiment results because of the similarity in transformed images to the different images in the datasets. Besides the style of writing, digits seem analogous to other images.

Overall, the proposed adapted models achieved high accuracy values reached to 99% with significant recall and precision values on the ten digits with 99% and standard deviation reached on the average to $\pm$ 0.01. In comparison to the state-of-the-art techniques, our adapted models showed a comparable and even in some case effective results ranged approximately from 98% up to 99.3%. In future work, we are planning to extend the experiments to cover other transfer deep learning models and to address more images by tuning the data augmentation settings. In addition, a variety of Arabic datasets are being deployed for comparison perspectives.

In future work, we plan to expand the experiments to include the other transfer learning architectures and examine their effectiveness as hybrid adaptive models. We also plan to optimize a variety of hyper-parameters on the most effective models with an increasing number of images using data augmentation technique. In particular, a random search is being used to select the number of epochs and learning rate to mitigate the problem of overfitting by getting over fitted parameters (Bergstra and Bengio 2012). Finally, due to the influence the time complexity on the efficiency of transfer models, we will present the time complexity of the adaptive models to examine their efficiency.

Compliance with ethical standards

Conflict of interest The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

Funding This work is not funded.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Abdleazeem S, El-Sherif E (2008) Arabic handwritten digit recognition. IJDAR 11(3):127–141

AlKhateeb JH, Alseid M (2014) DBN-based learning for arabic handwritten digit recognition using DCT features. In: 2014 6th International conference on computer science and information technology (CSIT), pp 222–226. IEEE

Alkhawaldeh RS (2019) DGR: gender recognition of human speech using one-dimensional conventional neural network. Sci Program. https://doi.org/10.1155/2019/7213717

Alkhawaldeh RS, Khawaldeh S, Pervaiz U, Alawida M, Alkhawaldeh H (2019) NIML: non-intrusive machine learning-based speech quality prediction on VoIP networks. IET Commun 13(16):2609–2616

Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, Hasan M, Van Essen BC, Awwal AA, Asari VK (2019) A state-of-the-art survey on deep learning theory and architectures. Electronics 8(3):292

Bengio Y et al (2009) Learning deep architectures for ai. Foundations and trends® in Machine Learning 2(1):1–127

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281–305

Canziani A, Paszke A, Culurciello E (2016) An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678

Chollet F (2018) Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek. MITP-Verlags GmbH & Co KG, Wachtendonk

Dargan S, Kumar M, Garg A, Thakur K (2019) Writer identification system for pre-segmented offline handwritten devanagari characters using k-NN and SVM. Soft Comput. https://doi.org/10.1007/s00500-019-04525-y

Elleuch M, Maalej R, Kherallah M (2016) A new design based-SVM of the CNN classifier architecture with dropout for offline arabic handwritten recognition. Procedia Comput Sci 80:1712–1723. https://doi.org/10.1016/j.procs.2016.05.512

Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

Hochreiter S (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int J Uncertain Fuzziness Knowl Based Syst 6(02):107–116

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

Khawaldeh S, Pervaiz U, Rafiq A, Alkhawaldeh RS (2018) Noninvasive grading of glioma tumor using magnetic resonance imaging with convolutional neural networks. Appl Sci 8(1):27

Kim B, Yuvaraj N, Ramasamy S, Santhosh R, Sabari A (2020) Enhanced pedestrian detection using optimized deep convolution neural network for smart building surveillance. Soft Comput. https://doi.org/10.1007/s00500-020-04999-1

Krizhevsky A, Sutskever I, Hinton G.E (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436

LeCun Y, Bottou L, Bengio Y, Haffner P et al (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324

Loey M, El-Sawy A, El-Bakry H (2017) Deep learning autoencoder approach for handwritten arabic digits recognition. arXiv preprint arXiv:1706.06720

Maheshwari M, Namdev D, Maheshwari S (2018) A systematic review of automation in handwritten character recognition. Int J Appl Eng Res 13(10):8090–8099

Mahmoud S (2008) Recognition of writer-independent off-line handwritten Arabic (Indian) numerals using hidden Markov models. Sig Process 88(4):844–857. https://doi.org/10.1016/j.sigpro.2007.10.002

Mahmoud SA (2008) Arabic (Indian) handwritten digits recognition using Gabor-based features. In: 2008 International conference on innovations in information technology, pp 683–687. IEEE

Mudhsh M, Almodfer R (2017) Arabic handwritten alphanumeric character recognition using very deep neural network. Information 8(3):105

Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359

Perez L, Wang J (2017) The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621

Rawat W, Wang Z (2017) Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput 29(9):2352–2449

Rosa GH, Papa JaP, Yang XS (2018) Handling dropout probability estimation in convolution neural networks using meta-heuristics. Soft Comput 22(18):6147–6156. https://doi.org/10.1007/s00500-017-2678-4

Salameh M (2014) Arabic digits recognition using statistical analysis for end/conjunction points and fuzzy logic for pattern recognition techniques. World Comput Sci Inf Technol J 4(4):50

Selvi P.P, Meyyappan T (2013) Recognition of Arabic numerals with grouping and ungrouping using back propagation neural network. In: 2013 International conference on pattern recognition, informatics and mobile engineering, pp 322–327. IEEE

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826

Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E (2018) Deep learning for computer vision: a brief review. Comput Intell Neurosci. https://doi.org/10.1155/2018/7068349

Younis KS (2017) Arabic handwritten character recognition based on deep convolutional neural networks. Jordanian J Comput Inf Technol 3(3):186