Egyptian Informatics Journal 23 (2022) 373–382

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Writer adaptation for E2E Arabic online handwriting recognition via adversarial multi task learning

Fakhraddin Alwajih [a,b,*], Eman Badr [a,c], Sherif Abdou [a]

[a] Department of Information Technology, Cairo University, Giza, Egypt
[b] Department of Computer Science and Information Technology, Ibb University, Ibb, Yemen
[c] University of Science and Technology, Zewail City of Science, Technology and Innovation, Giza, Egypt

ARTICLE INFO

Article history:
Received 20 September 2021
Accepted 12 February 2022
Available online 1 March 2022

Keywords:
Writer adaptation
Adversarial learning
Multi task learning
Arabic online handwriting recognition
Connectionist temporal classification
Convolutional neural networks
Bidirectional long short-term memory

ABSTRACT

The importance of online handwriting recognition has been rapidly increasing over recent years due to the rapid technological advances in handheld devices and communication software with handwriting interfaces. Deep learning end-to-end (E2E) models have provided high recognition rates as part of online handwriting recognition systems. However, attaining even higher performance levels requires supplementing these models with adaptation techniques that cater to individual penmanship. This study proposes a writer adaptation technique for Arabic online handwriting recognition systems that employs adversarial Multi-Task Learning (MTL). Adversarial training and MTL modify the deep-features distribution of the Writer Dependent (WD) model, leading its output to closely resemble that of the Writer Independent (WI) model. The design of the proposed method entails two tasks: label classification (primary task) and model features discrimination (secondary task). Our method was designed to jointly optimize both sub-networks. The proposed technique was tested against the E2E Connectionist Temporal Classification (CTC) based model, a combination of both Convolutional Neural Networks (CNNs) and Bidirectional Long Short-term Memory (BiLSTM). The proposed models were trained and evaluated against two large datasets (the Online-KHATT and CHAW). In supervised adaptation, it achieved an absolute Character Error Rate (CER) of up to 1.83% and an absolute Word Error Rate (WER) reduction of 11.71% over the WI model. Additionally, supervised adaptation achieved an absolute CER of up to 0.84% and an absolute WER reduction of 6.77% over the fine-tuned model. In unsupervised adaptation, the proposed method achieved an absolute CER of up to 0.5% absolute and an absolute WER reduction of 1.74% absolute (WER) reduction over the WI. Our experimental results indicate that our proposed supervised writer adaptation can achieve significant improvements in recognition accuracy compared with the baseline models: WI and fine-tuned models.

1. Introduction

The importance of handwriting recognition technologies has been rapidly increasing over recent years, due in part to the continuous advancements in both hardware and software. Most smartphone and tablets are now equipped with pen input enabled interfaces. Handheld devices have also become more popular and affordable to both students and businesses. Various applications such as note-taking, math typing, and drawing tool applications have found a growing market. Most videotelephony applications have added options for note-taking and handwritten annotation using digital pen capabilities.

Handwriting recognition essentially involves the transcription of handwritten input into digital text. Based on the nature of the

* Corresponding author at: Department of Information Technology, Cairo University, Giza, Egypt.
E-mail addresses: f.alwajih@grad.fci-cu.edu.eg (F. Alwajih), emostafa@zewailcity.edu.eg (E. Badr), s.abdou@fci-cu.edu.eg (S. Abdou).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.eij.2022.02.007
1110-8665/

input, handwriting recognition systems can be classified into either online or offline systems. In online handwriting recognition systems, input is captured through the use of a digital pen or stylus. Input format is a sequence of points. Additional information, based on the type of capturing device, may include timestamps and pressure. Due to the temporal nature of the input, recognition of online handwriting can be streamed. On the other hand, in offline handwriting systems, input is in the form of an image with width and height. Sources of offline input include scanned documents and images extracted from online input. As a result of the nature of offline input, handwriting recognition in offline systems cannot be streamed. In addition, treatment of input from online and offline handwriting during the preprocessing and features extraction stage is dissimilar. Whilst online handwriting is typically treated as a signal or time series, offline handwriting is treated as an image.

Approaches to online handwriting recognition can be divided into three main categories; segment and decode [18]; prototype-based classifiers [16,9], and sequence modeling [8]. Sequence modeling is the most common approach. A number of techniques have been utilized as part of sequence modeling, including Hidden Markov Models (HMMs) [23], hybrid Neural Networks (NNs) with HMMs (NNs/HMMs) [6], hybrid deep neural networks (DNNs) with HMMs (DNNs/HMMs) [19], and end-to-end (E2E) DNNs [24,8].

Traditional approaches to sequence modeling, including HMMs and hybrid approaches, typically include multiple components that need to be trained separately, leading to suboptimality. Deep learning E2E systems have allowed for the simultaneous training of all components of the model in an E2E fashion. Various deep learning models of different architectural designs have previously been utilized in the online handwriting recognition task, including CTC-based and Sequence to Sequence (seq2seq) models. CTC-based systems [24] have been used to recognize and map online handwriting input frames into letters, sub-units, or words. In this study, a CTC-based model was used in the design of the proposed method. CTC-based models can be incorporated with a Language Model (LM), allowing for improved functionality [8]. For a detailed review, please refer to [3,36,29].

E2E online handwriting recognition systems need to be trained using large datasets of thousands of handwriting samples. E2E systems have shown encouragingly comparable results to those attained through traditional systems [4,8]. Their performance, however, typically degrades when a WI model is tested against the handwriting of an unknown writer. The degradation is attributable to a mismatch between the distributions of training samples and the 'unseen' samples of a specific person. Differences in writing styles and devices used may also contribute to such a mismatch. Arabic handwritten script has been traditionally classified into six main styles: Naskh, Thuluth, Nastaliq, Dewani, Reqaa, and Kufi. Fig. 1 illustrates these six styles of traditional Arabic handwritten script. Some of these styles have been more commonly used in writing than others. Moreover, modern handwritten Arabic script typically shows elements of a number of learned traditional styles, leading to unpredictability. This unpredictability adds another challenge to the design of an effective Arabic handwriting recognition system.

One solution to reduce the mismatch between training data and unseen samples of a specific writer is to modify the model into a WD model. Writer adaptation is a process of converting the WI model to the WD model, given adaptation data from a specific writer. The main challenge to writer adaptation is having limited data. A WI model with a large number of parameters can easily get overfitted if adapted with limited adaptation data. Adaptation in deep learning is closely related to such techniques as domain adaptation, transfer learning and fine-tuning.

Few approaches have been proposed for writer adaptation in online handwriting recognition systems. They could be classified into two categories: model-based and features transformation-based writer adaptation methods [26,28,20,2,7,16,34,35]. Model-based adaptation methods rely on updating model parameters by optimizing a certain criterion. To that end, a number of techniques have been used, including fine-tuning, linear transformation, regularization and subspace methods. On the other hand, features transformation writer adaptation methods have typically depended on features space and typically function through either transforming input features or using auxiliary features [32]. The objective of these methods is to normalize variations in a writer's handwriting.

Writer adaptation for online handwriting models may be utilized in two different settings: supervised and unsupervised learning settings. In the supervised mode, adaptation samples are known to be labeled, while in the unsupervised model, adaptation samples are initially unlabeled and labeling is achieved through the WI model before adaptation is applied.

In this paper, we propose a novel writer model-based adaptation method that utilizes a CTC-based model to achieve optimal Arabic online handwriting recognition. Our method employs adversarial multi-task learning (MTL). Adversarial MTL techniques alleviate deep features mismatch between distributions of features generated by the WI model and the WD model. In adversarial learning, two tasks are involved: a primary task of label classification and a secondary task deep features discrimination. We evaluated the proposed method using two datasets; the CHAW and Online-KAHTT datasets. The proposed method has both consistently and significantly shown superior performance in comparison to the WI and fine-tuned WI models with the same data of target writers.

Contributions made by this work can be summarized as follows: (1) We introduce a new CTC-based method of adaption online handwriting recognition of a specific writer using adversarial multi-task learning and a small number of adaptation samples. (2) We developed a CTC-based model utilizing CNNs and BiLSTMs for Arabic online handwriting recognition, where we utilized two large datasets for training purposes. (3) We evaluated the proposed method using samples of writers from two datasets and benchmarked it against the fine-tuning method.

The paper is organized as follows: Section 2 explores previous work done in writer adaptation for online handwriting recognition. In Section 3, we explain the architecture of our proposed method. Section 4 details the results and provides a comparison of the proposed architecture against the original WI with both supervised and unsupervised adaptation settings. Lastly, Section 5 details our conclusions and recommendations for possible future work.

2. Related work

Model-based adaptation involves modifying and altering the model parameters or adding new parameters to the model to tune the pre-trained model to a target writer, with only limited adaptation data belonging to the specific writer.

Several approaches to model adaptation have been proposed. In [26], Matic et al. designed a writer dependent system based on a pre-trained Time-Delay Neural Network (TDNN) character-based recognition system. Adaptation is conducted by altering the last layer of TDNN, which acts as an optimal hyperplane classifier. This last layer is retrained to a new writing style. In [28], authors placed an Output Adaptation Module (OAM) on top of WI neural networks. OAM is a radial basis function (RBF) network that maps the output of the WI module into a writer-adapted output.

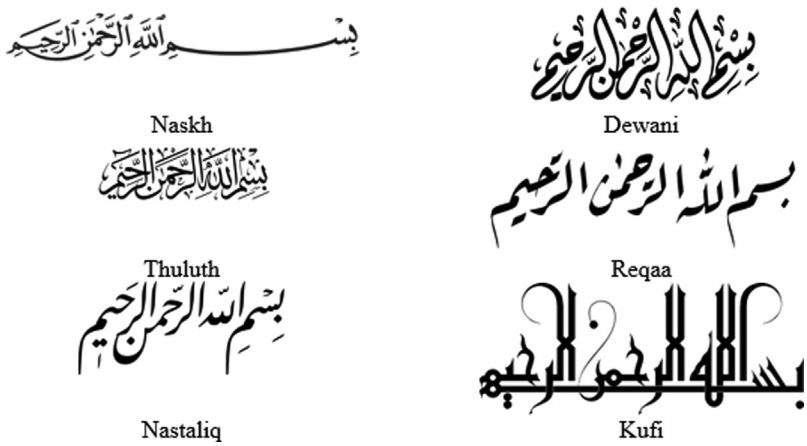

Fig. 1. Traditional Arabic handwriting styles.

The use of regularization can also aide in writer adaptation. In [20], researchers proposed a writer adaptation method for Support Vector Machines (SVMs) using biased regularization. Adaptation is accomplished by minimizing an adjusted regularized risk functional on adaptation data. Biased regularization provides a trade-off between WI and WD (generic and personalized information). As HMMs had provided remarkable results in online handwriting recognition, some studies utilized adaptation methods from speaker adaptation in Automatic Speech Recognition (ASR) [2]. In [7], writer adaptation was augmented with a Maximum Likelihood Linear Regression (MLLR) for a HMMs based recognition system. In that approach, MLLR 'attempts' to find the best transformation matrix that would maximize the likelihood of the adaptation data. The transformation matrix can then be applied to the HMMs state observation densities to adjust the WI model to the new writer's data.

Features transformation adaptation, on the other hand, relies on applying transformations to reduce the variations between individual writers' data. Transformations can be either applied directly to the features vector or raw handwriting input or could function to extract deep independent features (writer's code), subsequently incorporating these features into the recognition phase. Huang et al. [16] utilized Linear Discriminant Analysis (LDA) for writer adaptation of prototype-based methods. They proposed an Incremental LDA (ILDA) algorithm for writer adaptation. In their method, the ILDA algorithm was designed to adjust both the LDA transformation matrix and the prototypes.

Zhang et al. [37] proposed learning a Style Transfer Mapping (STM) paradigm that conducts computations for each individual writer. In STM, the source point set, defined as writer-specific data, is mapped and transformed towards the target point set, defined as the corresponding parameters for the WI classifier. This is achieved through solving a convex quadratic programming problem with the closed-form solution. STM reduces style variations between different writers by projecting their writing onto a style-free space. Consequently, a WI classifier is able to achieve a higher recognition rate. STM can be described as features transformation that is applied in either a supervised, unsupervised or semi-supervised manner. As an extension for STM, Yang et al. [34] proposed complementing the paradigm with an integrated framework for neural network classifiers. In the adaptation, they added a layer to the WI classifier. The adaptation layer weights were updated, and WI model weights were kept fixed. The authors demonstrated that the proposed model outperformed other traditional STM methods, including Learning Vector Quantization (LVQ) and the Modified Quadratic Discriminant Function (MQDF).

Yang et al. [35] proposed four types of transformations that they specifically designed for CNN layer adaptation. They proposed that adaptation could be performed deeply on multiple layers of the model, including the CNN and fully connected layers, rather than exclusively on the top layers. The authors referred to their method as Deep Transfer Mapping (DTM), entirely run in an unsupervised manner. Adaptation is achieved through the capturing of additional information from different abstraction levels. During the adaption procedure, adaptation layers serve as the linear transformation.

Deep learning methods were also utilized in features transformation adaptation. In [9], researchers used deep learning features to perform writer adaptation for online Chinese handwriting recognition. In this method, DNNs and CNNs are utilized to extract these features (called tandems). Extracted features are then fed into the prototype-based classifier. A linear transformation of the deeply learned features is used to perform adaptation.

Designers of the aforementioned writer adaptation methods for online handwriting recognition systems primarily utilized traditional techniques such as HMMs, SVM, TDNN and prototype classifiers. Despite that a small number of these methods did incorporate deep learning in their features representation, none of their creators tackled the use of E2E model-based writer adaptation. Languages addressed were primarily either Latin or Chinese languages.

E2E online handwriting systems have recently provided remarkable recognition rates [8] and have become the dominant online handwriting systems in the field. E2E systems rely on deep learning neural networks such as BiLSTMs and CNNs. Most of these systems utilize the CTC algorithm in the training process. CTC allows training on seq2seq problems. There has been extensive research into domains of E2E ASR [22,36,27] and optical character recognition (OCR) systems [33,17] which are closely related to online handwriting recognition. In [22], they investigated two-speaker adaptation methods, the Kullback-Leibler Divergence (KLD) and MTL adaptation, using an E2E ASR CTC-based model. Adversarial based adaptation methods are mainly imported from the domain adaptation area [10,11]. To the best of our knowledge, our proposed model is the first E2E online handwriting recognition model that examines writer adaptation for Arabic online handwriting. In this work, we introduce a regularization technique as a model-based writer adaptation for online handwriting recognition task using adversarial learning.

3. Writer Adaptation for End-to-End CTC based models

In this work, we propose a novel model-based writer adaptation method utilizing a CTC-based model and an adversarial MTL. In adversarial learning, a primary task of label classification and a secondary task of deep features discrimination are defined. WD

is created and initialized by cloning WI, then divided into features extraction and label classification sub-networks. In the secondary task, a discriminative sub-network is attached on top of the features extraction sub-network in parallel with a label classification sub-network of the primary task. The two tasks are jointly optimized in a minimax manner with multi-task learning, where features extraction layers are shared among the tasks. The goal of a minimax game is to minimize label classification loss while jointly maximizing features discrimination loss. By the end of this game, the regularized WD model should reach a higher recognition rate against the test data of the target writer than that of the WI model. The WI is kept fixed during adaptation and discarded with the discriminative sub-network during the test.

In the following section, we describe the CTC-based model used in this work, as well as the Generative Adversarial Networks (GANs) and Domain Adaptation Neural Networks (DANNs). We then describe the writer adversarial training for online handwriting writer adaptation.

3.1. CTC based online handwriting recognition system

We used a combination of CNN, BiLSTM and fully connected layers to build our model architecture, as shown in Fig. 2. During training, CTC loss is utilized as an objective function [14,24] for the optimization of the prediction of the labels' sequence. Typically, the length of the output sequence for handwriting recognition is less than the length of the handwriting input. In CTC, an additional blank label e is introduced to ensure that the length of outputs remains the same, and to allow for repeated labels in the output.

The CTC objective function L_{CTC} is defined as the summation of negative log conditional probabilities of the correct labels, as shown in Eq. 1 below. In this summation, $X = x_1, x_2, .., x_T$, where X is the handwriting input sequence with length $T, Y = l_1, l_2, \ldots, l_L$ is denotes the label sequence with length $L, Theta$ denotes model weights, π denotes the CTC single path, and $B^{-1}(L)$ denotes as all possible CTC paths generated from original L labels' sequence.

$$L_{CTC} = -lnP_\Theta(Y|X) = -ln \sum_{\pi \in B^{-1}(L)} P_\Theta(\pi|X) \quad (1)$$

The CTC assumes that output units are conditionally independent. Hence, $P_\Theta(\pi|X)$ can be expressed as the product of posteriors from each time step t:

$$P_\Theta(\pi|X) = \prod_{t=1}^{T} P_\Theta(\pi_t|x_t) \quad (2)$$

where π_t is the output unit at time t, x_t is the input handwriting at time t. The posterior probability of the labels at each time step is computed using CNN-BiLSTM deep network (*encoder*). The encoder outputs $h_{enc} = (h_{enc1}, .., h_{encT})$, are treated as logits and are passed to the softmax layer to compute the probability distribution over labels l. An estimation of model parameters Θ is then made by maximizing Eq. 1 using gradient descent.The gradients required in gradient descent can be calculated using the forward-backward algorithm [15].

To find the most probable sequence of labels given a handwriting input, greedy decoding is utilized [14]. Greedy decoding involves two steps. The first step is concatenating the most probable label for every time step. The second step is removing the duplicate labels and all blanks.

3.2. Generative Adversarial Networks (GANs)

GANs are a form of neural network architecture that allows deep learning models to learn and capture the training data distribution, allowing for the generation of new data instances from learned distribution [12]. Typically, GANs consist of two models; a discriminator model and a generator model, as shown in Fig. 3. The discriminator is a binary classifier that functions to distinguish real instances from fake instances generated by the generator. The generator is, on the other hand, the part of GANs that generates fake instances with the same distribution as that of the training data.4.

Assuming that x is a sample instance that represents input data, then $D(x)$ is the discriminator. $D(x)$ is a binary classifier that produces a probability with x as a real instance from the training data. $D(x)$ should output a high probability with respect to real instances and a low probability with respect to synthesized instances. Similarly, assuming that z is a latent vector sampled from Gaussian distribution, then $G(z)$ is the generator that transforms z to the data space.

The main aim of G is to produce fake samples from estimated distribution (p_g) by capturing the training data distribution(p_{data}). According to [12], G and D play a minimax game. G attempts to minimize the probability that D would classify its synthesized out-

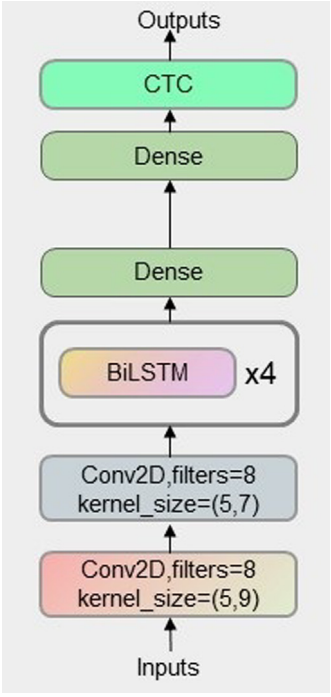

Fig. 2. The architecture of the E2E system is used to train Arabic online handwriting recognition. The proposed CTC-based model consists of CNN layers followed by BiLSTM layers and fully connected layer.

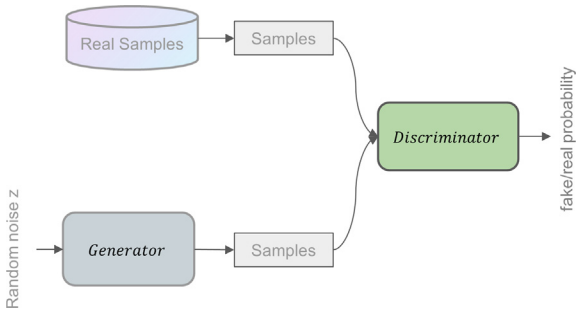

Fig. 3. General architecture of Generative Adversarial Networks (GANs).

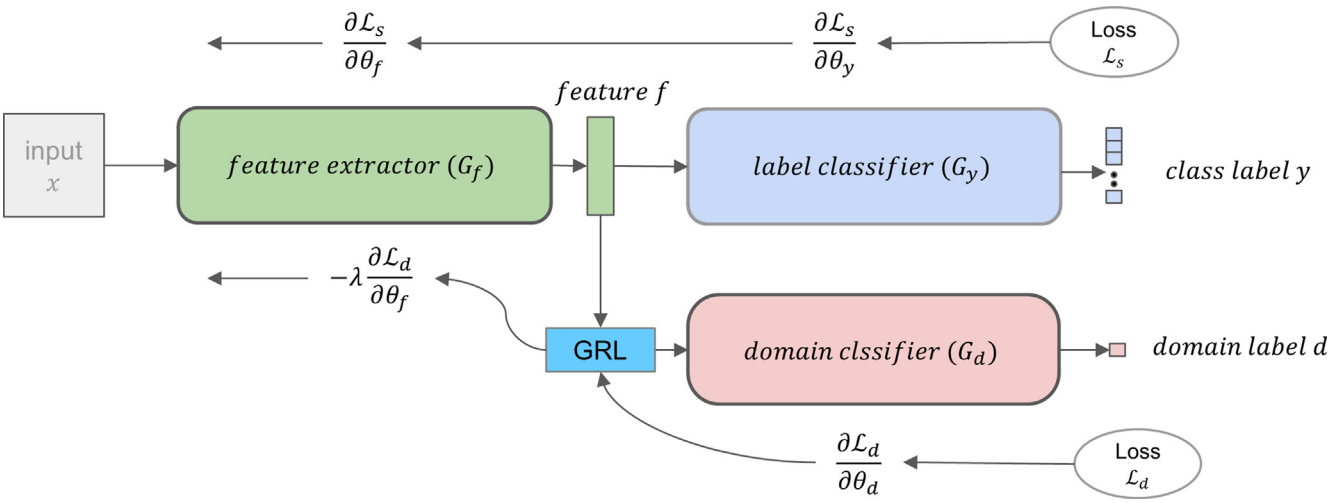

Fig. 4. The general architecture of DANN is composed of three main sub-networks: a features extractor sub-network (G_f), a label classifier (G_y) sub-network, and a domain classifier (G_d) sub-network. The figure was adapted from [11].

puts as fake instances ($log(1 - D(G(x)))$), whilst D attempts to maximize instances of correctly classifying synthesized and realistic samples ($logD(x)$). According to [12], the GAN loss function may be expressed as follows:

$$\begin{aligned}\min_G \max_D V(D,G) &= \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] \\ &+\mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]\end{aligned} \tag{3}$$

Theoretically, the optimal solution for this minimax competition is attained when the estimated distribution by the generator G equals the training data distribution, and the discriminator cannot distinguish between real and fake inputs. In practice, GANs models are not trained up to this point.

3.3. Domain Adaptation Neural Networks (DANNs)

DANNs are deep domain adaptation networks that mainly rely on the representation learning (RL) technique, where both domain invariance and discriminative properties are considered during the learning process [11].

The goal of DANNs is to learn a classifier ϕ , which can be broken down into two parts, $\phi = G_f \circ G_y$, where $G_f$is a features extractor that represents underlying features and $G_y$is a label classifier that predicts the target labels. Both $G_f$and G_y are used during training and testing and optimized by minimizing the following equation:

$$\mathscr{L}_y(\theta_f,\theta_y) = \frac{1}{N_s}\sum_{(x,y)\in D_s} \ell(G_y(G_f(x)),y) \tag{4}$$

In addition, DANNs architecture proposes a domain classifier G_d, which is branched after features representation G_f to distinguish between the target and source domains. G_d is used during the training phase and thrown after training. It is optimized jointly with G_f and G_y by maximizing the following function:

$$\begin{aligned}\mathscr{L}_d(\theta_f,\theta_d) &= \tfrac{1}{N_s}\sum_{x\in D_s}\ell(G_d(G_f(x)),s) \\ &+\tfrac{1}{N_t}\sum_{x\in D_t}\ell(G_d(G_f(x)),t)\end{aligned} \tag{5}$$

Total training loss may be expressed as follows:

$$E(\theta_f,\theta_y,\theta_d) = \mathscr{L}_y(\theta_f,\theta_y) - \lambda\mathscr{L}_d(\theta_f,\theta_d) \tag{6}$$

where λ is the trade-off between $\mathscr{L}_y$ and $\mathscr{L}_d$. The parameters θ_f, θ_y and θ_d are obtained by optimizing the following equations:

$$(\hat{\theta}_f,\hat{\theta}_y) = \underset{(\theta_f,\theta_y)}{\operatorname{argmin}}\ E(\theta_f,\theta_y,\hat{\theta}_d) \tag{7}$$

$$(\hat{\theta}_d) = \underset{(\theta_d)}{\operatorname{argmax}} E(\hat{\theta}_f,\hat{\theta}_y,\theta_d) \tag{8}$$

Applying stochastic gradient descent or one of its variants, the above DANN objective yields the following updating rules:

$$\theta_f = \theta_f - \alpha\left(\frac{\partial \mathscr{L}_y}{\partial \theta_f} - \lambda\frac{\partial \mathscr{L}_d}{\partial \theta_f}\right) \tag{9}$$

$$\theta_y = \theta_y - \alpha\frac{\partial \mathscr{L}_y}{\partial \theta_y} \tag{10}$$

$$\theta_d = \theta_d + \alpha\frac{-\lambda\partial \mathscr{L}_d}{\partial \theta_d} \tag{11}$$

As a typical training phase in NNs, gradients are calculated by differentiating losses with respect to parameters using standard back-propagation. In DANNs, the authors in [11] proposed GRL for the backward pass.

3.4. Writer adaptation using Adversarial Training with MTL

To perform the writer adaptation task, we start with a sequence of features $X = x_1, x_2, \ldots, x_T$, target labels $Y = y_1, y_2, \ldots, y_L$,and a trained WI model Θ^{WI}. As shown in Fig. 2, we spilt WI into two sub-networks G_f^{WI} and G_y^{WI}. G_f^{WI} acts as a features extractor, mapping each input vector to its representation. G_y^{WI}sub-network acts as a classifier to predict posterior probabilities of classes given the representation from G_f^{WI}. We then construct our WD model by copying the WI model with the same architecture. As a result, we have $G^{WD} = G_f^{WD}.G_y^{WD}$ initialized from Θ^{WI}. G^{WD} architecture is trained through the minimization of the following CTC loss objective:

$$L_{\text{CTC}}(\Theta_f^{WD},\Theta_y^{WD}) = -\log\left(\sum_{\pi\in\mathscr{B}^{-1}(\mathbf{L})} p(\Theta_f^{WD},\Theta_y^{WD})(\pi|\mathbf{x})\right) \tag{12}$$

$$p(\Theta_f^{WD},\Theta_y^{WD})(\pi|\mathbf{x}) = \prod_{t=1}^{T} p(\Theta_f^{WD},\Theta_y^{WD})(\pi_t|\mathbf{x_t}) \tag{13}$$

An important goal is 'force' the output of G_f^{WD} to be closely resemble the output of G_f^{WI} for adaptation samples of the target writer. To reach this goal, an additional discriminator binary classifier G_d is introduced. G_d accepts input form both G_f^{WD} and G_f^{WI}.

Adversarial training of GW D and Gd is then conducted. During this training, L_{disc} is minimized with respect to θ_d and maximized L_{disc} with respect to θ_f^{WD}. This minmax objective rivalry would improve the ability of G_f^{WD} to produce features with a distribution closer to the features distribution produced by G_f^{WI}, and would improve the discriminator capability of G_d, allowing it to better to distinguish between features generated by G_f^{WI} versus G_f^{WD}.

Eventually, the minimax competition would converge to the point where G_f^{WD} produces highly perplexing features that G_d is unable to differentiate whether these features are produced by G_f^{WD} or G_f^{WI} for the same sample. Reaching convergence, the WD model would effectively be regularized and would have an excellent capability to generalize with respect to target writer data, without diverging too much from the WI model. G_d is trained by maximizing the standard cross-entropy loss as follows:

$$\begin{aligned}\mathscr{L}_{disc}(\Theta_f^{WD},\Theta_f^{WI},\Theta_d) &= \tfrac{1}{T}\sum_{t=1}^{T}\ell(G_d(G_y^{WD}(x_t));\Theta_f^{WD},\Theta_d)\\ &+\tfrac{1}{T}\sum_{t=1}^{T}\ell(G_d(G_y^{WI}(x_t));\Theta_f^{WI},\Theta_d)\end{aligned} \tag{14}$$

In this context, WD and G_d models are jointly trained through adversarial MTL training. In the MTL setting, there is the primary task of label classification and the secondary task of featuress discrimination. Features discrimination functions to predict if features are produced by G_f^{WD} or G_f^{WI}. The primary and the secondary tasks are optimized adversarially using the following objective function:

$$\begin{aligned}\mathscr{L}_{total}(\Theta_f^{WD},\Theta_y^{WD},\Theta_d) &= \mathscr{L}_{CTC}(\Theta_f^{WD},\Theta_y^{WD})\\ &-\lambda\mathscr{L}_{disc}(\Theta_f^{WD},\Theta_f^{WI},\Theta_d)\end{aligned} \tag{15}$$

$$\hat{\Theta}_f^{WD},\hat{\Theta}_y^{WD} = \arg\min \mathscr{L}_{total}(\Theta_f^{WD},\Theta_y^{WD},\hat{\Theta}_d) \tag{16}$$

$$\hat{\Theta}_d = \arg\min \mathscr{L}_{total}(\hat{\Theta}_f^{WD},\hat{\Theta}_y^{WD},\Theta_d) \tag{17}$$

where λ is a hyperparameter which controls the trade-off between L_{CTC} and L_{disc} during the optimization, $\hat{\Theta}_f^{WD}$, $\hat{\Theta}_y^{WD}$, and $\hat{\Theta}_d$ are parameters to be optimized. Θ_f^{WI} is used to generate the features and set to be fixed during the training process along with Θ_y^{WI}. In addition, $\hat{\Theta}_f^{WD},\hat{\Theta}_y^{WD},\hat{\Theta}_d$ parameters are updated during training via back-propagation algorithm with gradient descent, expressed as follows:

$$\theta_f^{WD} \leftarrow \theta_f^{WD} - \alpha\left(\frac{\partial\mathscr{L}_{CTC}}{\partial\theta_f^{WD}} - \lambda\frac{\partial\mathscr{L}_{disc}}{\partial\theta_f^{WD}}\right) \tag{18}$$

$$\theta_y^{WD} \leftarrow \theta_y^{WD} - \alpha\frac{\partial\mathscr{L}_{CTC}}{\partial\theta_y^{WD}} \tag{19}$$

$$\theta_d = \theta_d + \alpha\frac{-\lambda\partial\mathscr{L}_{disc}}{\partial\theta_d} \tag{20}$$

where α is the learning rate. GRL then allow for adversarial learning in the back-propagation phase. Upon completion of training, both the secondary task network G_d and fixed WI network are discarded. We keep the adapted primary task network for testing.

While the architecture of the primary task is sequence to sequence, the secondary task model architecture is a sequence to one. As CTC deals with mapping an input sequence to an output sequence in the primary task, we added max and average pooling layers to the discriminator network. These layers aggregate the variable-length hidden vector into two fixed hidden vectors to the next dense layer.

4. Experiments and results

We performed writer adaptation for an Arabic online handwriting recognition task. We utilized two large datasets for training and testing of our end-to-end CTC-based model. For the adaptation task, individual writer samples were chosen from the test sets.

4.1. Datasets

We trained and validated the WI and WD models using two well-known Arabic online handwriting datasets; the CHAW [19] and Online-KHATT [25] datasets. The CHAW comprised data collected from 1250 writers and was proposed by Cairo University, whilst the Online-KHATT comprised data collected from 623 writers and was proposed by King Fahd University; a total of 1873 writers from both datasets. Writers varied in gender, country of origin, age, handedness, and level of education. Both datasets contained natural writing with unrestricted writing styles. This variation attests to the reliability of datasets. The test set for both datasets consisted of 240 writers. Table 1 compares and summarizes the both datasets.

4.2. Writer independent model setup

We used a CTC-based model with CNN and BiLSTM neural networks as the WI model for the Arabic online recognition system in our experiments. We chose 20 handwriting features described for each point described in [24,23,4] as the input features. Our design was included of two CNN layers followed by 4 BiLSTM layers with 80 units each. Finally, two fully connected layers were added at the point where the output layer had a total of 160 units, needed due to the nature of Arabic script. The Arabic alphabet is comprised of 28 letters shaped differently according to their position in a word. We thus expanded each letter into four distinct 'position-dependent letters' (isolated, start, middle, and end). This expansion resulted in 160 classes in the output layer. In decoding, we utilized the simplest greedy CTC decoder to select the highest probability over characters distribution. During the training phase (to increase network robustness), dropout implied dropping random neurons [30]. We utilized dropout in our WI model with a rate of 0.3 for all layers. Batch normalization (BN) is another technique used to accelerate the convergence of NNs and improve the capability of the network to generalize [5]. In our model, a BN layer was placed after each CNN layer in order to increase speed during training and attain better recognition accuracy during testing. WI was trained with CTC loss using Adam Optimizer [21]. TensorFlow 2.2 [1] was used for the purposes of building, training, and testing all models.

4.3. Writer adaptation model setup

For our writer adaption model, we used the writer adaptation method described in Section 3.4. and the WI trained model described in Section 4.2 in order to create the same architecture and initialize the weights of the WD model. The CNN, BiLSTM, and dense layers constituted the features extractor sub-network. The last dense layer constituted the label classifier sub-network, as shown in Fig. 5. The WI features extractor sub-network was used during the adaptation process to generate WI features from

Table 1
Comparison between CHAW and Online-KATHH datasets.

Dataset	CHAW		Online-KHATT	
	Train set	Test set	Train set	Test set
Number of writers	1146	104	623	136
Number of samples	180k	12k	6974	1533
Count of sentences	–	–	6974 (lines)	1533 (lines)
Count of words	180k	12k	56547	12100
Size of vocabulary	17800	500	19437	5887

adaptation samples. However, WI features extractor sub-network weights were kept fixed during the optimization process. We also defined the discriminator model (sub-network) as two BiLSTM layers with 64 units each, followed by average and max-pooling layers. A dense output layer with sigmoid activation was added. We attached the GRL layer to the front of the discriminator network, as illustrated by Fig. 6. The discriminator network was set to receive input from the WI features extractor sub-network and the WD features extractor sub-network and to attempt to distinguish between these outputs. As the structure of the discriminator model followed the sequence to one architecture, we employed a pooling layer in order to aggregate the output sequence of the BiLSTM layers - in both the WI and WD features extractor sub-networks - as a fixed vector and then pass it onto the dense output layer.

The dropout, which was used in the training WI model, was also utilized during adaptation training. BN layers weights were kept frozen during adaptation training. We employed early stopping regularization in order to prevent overfitting of the adapted models.

For supervised adversarial writer adaptation, we used the labels provided with handwriting samples in the datasets, while during unsupervised adversarial writer adaptation, we used the WI to label adaptation samples. We selected two writers from each dataset to assign and pick up the best hyperparameters for adaptation training. Random search was conducted to find the best hyperparameters (batch size, learning rate, dropout rate, and λ). The results of this assignment were then used for all writers.

4.4. Evaluation

For the purposes of evaluating our method, we used standard metrics character error rate (CER) and word error rate (WER). We used the same handwriting samples from the Online-KHATT and CHAW datasets (see Section 4.1) that were used to train the baseline model. Samples used in the adaptation process were not used in the training of the WI model.

For evaluation, selected handwritings of 84 writers from the test sets of both datasets; 42 writers from each dataset. For each writer, we split his/her samples into training and test sets. From the CHAW dataset, we selected 40 samples (words) for training and 80 samples for testing (to be used in all experiments). The online-KHATT dataset is comprised of sentence-based samples. From that dataset, we choose four samples per writer for adaptation training and an average of 10 samples for testing.

To evaluate our proposed adaptation method against other methods, we ran two additional experiments on a WI model and a fine-tuned model for each writer fine-tuned with the writer's training data. First, we ran WI for both datasets. Values of 13.21% CER, 41.50% WER, 16.84% CER and 56.05% WER were achieved

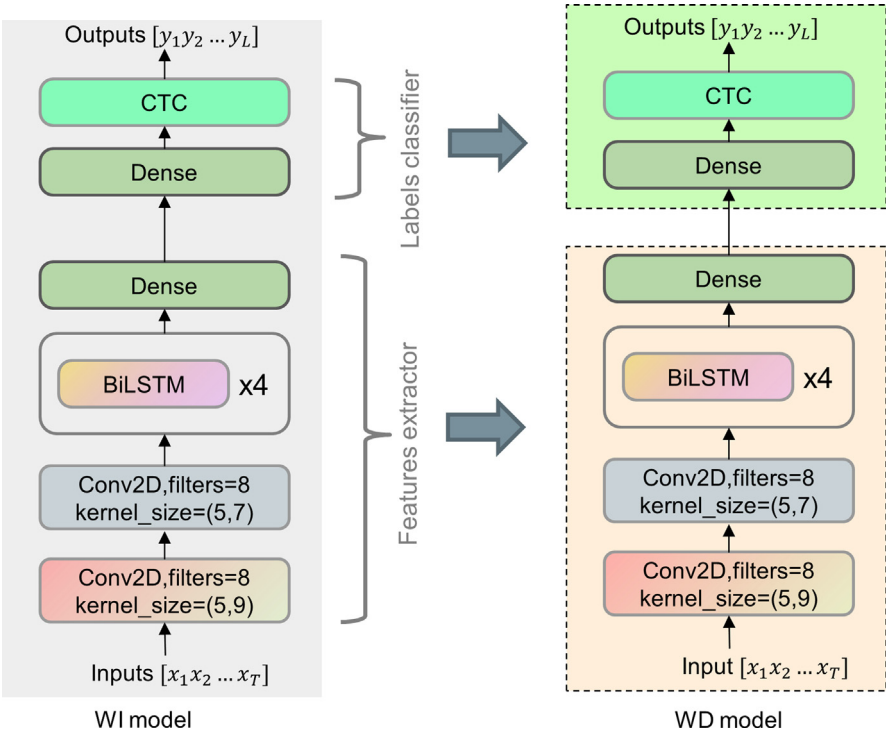

Fig. 5. To the left is the WI model, divided into a features extractor and label classifier. To the right is the WD model created and initialized from the trained WI model; also divided into a features extractor and label classifier.

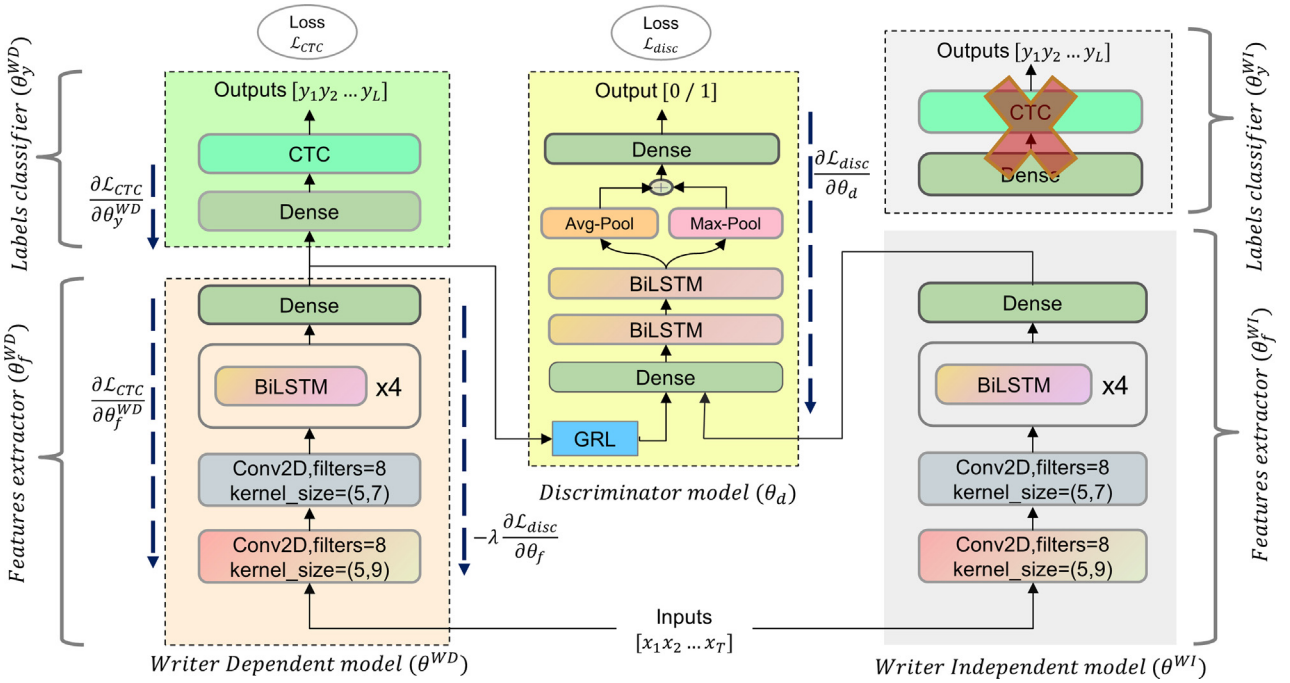

Fig. 6. The architecture of the adversarial writer adaptation. On the left is the Writer Dependent (WD) model θ^{WD} to be adapted. In the middle is the discriminator model θ_d that distinguishes between features generated by the θ_f^{WI} versus the θ_f^{WD} sub-networks. Finally, on the right is the Writer Independent (WI) model θ^{WI}, used to generate WI features of the input by θ_f^{WI}. The θ^{WD} model and discriminator θ_d are optimized adversarially, while the θ_f^{WI} is kept fixed during adaptation. The θ_y^{WI} sub-network is discarded during the training and the θ_f^{WI} and the θ_d are discarded after the training.

against the CHAW and Online-KHATT datasets, respectively. Similarly, we applied fine tuning to the WI model as a second baseline. Average result over all writers stood 12.77% CER, 39.62% WER 15.30% CER, 48.04% WER against the CHAW and Online-KHATT datasets, respectively (Table 2).

4.5. Discussion

An important finding during adaption involved the level of reductions in WERs. Reductions in WERs were higher than CERs both during adaptation (Table 2 and Fig. 7) and in the WI model. This may attributable to the fact that the WI model has the capability to recognize most of the target characters, but the result may contain some missed, misspelled or disordered words that cause higher WERs. Supervised adaptation helps in learning to overcome these errors and improves some word character ordering as well.

The Arabic script is comprised of 28 main characters, each of which is written differently according to their position in a word. In our CTC-based model, we expanded each charterer to 4 different 'position-based' characters (isolated, start, middle, and end). In addition, we added additional special Arabic characters such as the 'lam-alf,' as well as number figures and punctuation marks, creating a total of 160 units in the output layer. Expanding recognition of a character to include its shape in multiple positions contributes to the improvement of results compared with the use of the basic characters only. In Arabic script, character shape not only differs according to position in a word, but is also influenced by both the preceding and succeeding letters. Character with position carries contextual information which helps in the decoding phase [4]. Adaptation training data would not however cover all 160 target units for each individual writer. The proposed adaptation method works on features representation and performs the distribution shift based on the features detected. Thus, our model is able to satisfactorily generalize, despite possibly not covering all target units of a specific writer.

Fig. 7 details a comparison of the performance of the four models for each writer data from both datasets in terms of CER and WER. Evidently, the use of supervised and unsupervised adaptation constantly improved the recognition rate compared with WI and fine-tuned models for each writer. General robustness of proposed methods in relation to different handwriting styles was evident from most results. Two exceptions involved one writer from the CHAW dataset, and one from the Online-KHATT dataset. The writers wrote their writing samples with multiple lines, which limited the ability of the WI model to recognize letters/words. In both cases, adaptation was not effective.

Fine tuning gives better results than unsupervised adaptation. Unsupervised adaptation depends on the quality of the WI model

Table 2
The WER (%) and the CER (%) (lower is better) performance of WI, fine-tuned, supervised adversarial adaptation, and unsupervised adversarial adaptation models.

System	Dataset					
	CHAW		Online-KHATT			
	CER[%]	WER[%]	CER[%]	WER[%]	Avg. CER[%]	Avg. WER[%]
Writer Independent model	13.21	41.50	16.84	56.05	15.02	48.77
Fine tuning ($\lambda = 0$)	12.77	39.62	15.30	48.04	14.03	43.83
Supervised adversarial adaptation ($\lambda = 3$)	**11.97**	**35.22**	**14.42**	**38.90**	**13.20**	**37.06**
Unsupervised adversarial adaptation ($\lambda = 3$)	12.94	40.74	16.12	53.36	14.53	47.04

Fig. 7. Achieved CERs and WERs across different models (WI, fine-tuned, supervised adversarial adaptation, and unsupervised adversarial adaptation models) per writer for CHAW and Online-KHATT datasets. (a) CERs of CHAW dataset. (b) WERs of CHAW dataset. (c) CERs of Online-KHATT dataset. (d) WERs of Online-KHATT dataset.

which is used to generate labels for unlabeled adaptation data. In case of labels availability, it is recommended to go with supervised adaptation.

In all our experiments, we used greedy decoding for the model outputs. It selects the most probable character over characters distribution for each time step as described in Section 3.1. Using decoding techniques like beam search decoder with LM or dictionary integration could improve the recognition rate. In our study, we primally focused on improving upon model and the use of a greedy decoder that would be capable of highlighting improvements. Integration with LMs can be done after writer adaptation feasibility in the production systems.

Adaptation methods could improve baseline online handwriting recognition models. Moreover, investigating advanced sequence models like seq2seq with attention and self-attention transformers [31] can boost the result of baseline models before adaptations in

case of data availability. Then, we can apply adversarial multi-task learning adaptation to these models in the same manner as was done in this work. In addition, data augmentation is used to increase the training data. Thus, synthetic data generation [13] for writers with style transfer would increase the adaptation samples for writers, resulting in increased overall performance.

5. Conclusions

In this article, we presented a new method of model-based writer adaptation for CTC-based online handwriting recognition using adversarial multi-task learning. In this work, WD was cloned from WI and initialized by WI's weights. WD was then split into a features extractor and a label classifier. A GRL layer with a discriminator model was placed after the features extractor in parallel with the label classifier. Adversarial multi-task learning was used to ensure the distribution of deep features generated by the WD was similar to the distribution of deep features generated by the fixed WI. Experimental results indicate that the proposed supervised writer adaptation constantly improves the performance of adapted models over two baseline models; a WI and a fine-tuned model. In unsupervised writer adaptation, experimental results showed improvement over WI and degradation in comparison with fine-tuned models.

Our work is the first work that tackles writer adaptation for online handwriting recognition for E2E models. It will contribute to opening more doors for Arabic language applications which could be used in different domains. Also, The proposed method can be applied to other languages as well such as Latin languages. Adversarial multi-task learning adaptation is not limited to CTC-based models and it can be applied to other E2E models in the same manner such as attention and self-attention models. In the future, we will explore the possibility of applying adversarial multi-task learning on attention-based models. In addition, we will explore the effects of synthesized adaptation training data on the performance of adaptation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems..

[2] Abdelaziz I, Abdou S, Al-Barhamtoshy H. A large vocabulary system for arabic online handwriting recognition. Pattern Anal Appl 2016;19:1129–41.

[3] Al-Helali BM, Mahmoud SA. Arabic online handwriting recognition (aohr): A survey. ACM Comput. Surv. 2017;50.

[4] Alwajih F, Badr E, Abdou S, Fahmy A. Deeponkhatt: An end-to-end arabic online handwriting recognition system. Int J Pattern Recogn Artif Intell 2021.

[5] Amodei D, Ananthanarayanan S, Anubhai R, Bai J, Battenberg E, Case C, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. International conference on machine learning, PMLR 2016:173–82.

[6] Bengio Y, LeCun Y, Nohl C, Burges C. Lerec: A nn/hmm hybrid for on-line handwriting recognition. Neural Comput 1995;7:1289–303. doi: https://doi.org/10.1162/neco.1995.7.6.1289.

[7] Brakensiek A, Kosmala A, Rigoll G. Comparing adaptation techniques for online handwriting recognition. Proceedings of Sixth International Conference on Document Analysis and Recognition 2001:486–90. doi: https://doi.org/10.1109/ICDAR.2001.953837.

[8] Carbune V, Gonnet P, Deselaers T, Rowley HA, Daryin A, Calvo M, Wang LL, Keysers D, Feuz S, Gervais P. Fast multi-language lstm-based online handwriting recognition. International Journal on Document Analysis and Recognition (IJDAR) 2020:1–14.

[9] Du J, Zhai JF, Hu JS. Writer adaptation via deeply learned features for online chinese handwriting recognition. International Journal on Document Analysis and Recognition (IJDAR) 2017;20:69–78.

[10] Ganin Y, Lempitsky V. Unsupervised domain adaptation by backpropagation. In Proceedings of the 32nd International Conference on International Conference on Machine Learning – Volume 37, JMLR.org; 2015. pp. 1180–1189..

[11] Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, et al. Domain-adversarial training of neural networks. J Mach Learn Res 2016;17:2030–96.

[12] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S. Generative adversarial nets. In Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger KQ. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc.; 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf..

[13] Graves A. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850..

[14] Graves A, Fernández S, Gomez F, Schmidhuber J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning. p. 369–76.

[15] Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks. International conference on machine learning, PMLR 2014:1764–72.

[16] Huang Z, Ding K, Jin L, Gao X. Writer adaptive online handwriting recognition using incremental linear discriminant analysis. In: 2009 10th International Conference on Document Analysis and Recognition, IEEE. p. 91–5.

[17] Kang L, Rusinol M, Fornés A, Riba P, Villegas M. Unsupervised writer adaptation for synthetic-to-real handwritten word recognition. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. p. 3502–11.

[18] Keysers D, Deselaers T, Rowley HA, Wang LL, Carbune V. Multi language online handwriting recognition. IEEE Trans Pattern Anal Mach Intell 2017;39:1180–94.

[19] Khaled O, Fahmy A, Abdou S. Large vocabulary hybrid dnn/hmm arabic online handwriting recognition system. In: 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR). p. 876–81.

[20] Kienzle W, Chellapilla K. Personalized handwriting recognition via biased regularization. In: Proceedings of the 23rd international conference on Machine learning. p. 457–64.

[21] Kingma DP, Ba J. Adam: A method for stochastic optimization; 2014. arXiv preprint arXiv:1412.6980..

[22] Li K, Li J, Zhao Y, Kumar K, Gong Y. Speaker adaptation for end-to-end ctc models. 2018 IEEE Spoken Language Technology Workshop (SLT) 2018:542–9.

[23] Liwicki M, Bunke H. Hmm-based on-line recognition of handwritten whiteboard notes. In Proc. 10th Int. Workshop Frontiers in Handwriting Recognition; 2006. pp. 595–599..

[24] Liwicki M, Graves A, Bunke H, Schmidhuber J. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. Proc. 9th Int. Conf. on Document Analysis and Recognition 2007;1:367–71.

[25] Mahmoud S, Luqman H, Al-helali B, Binmakhashen G, Parvez M. Online-khatt: An open-vocabulary database for arabic online-text processing. Open Cybern Syst J 2018;12:42–59.

[26] Matic N, Guyon I, Denker J, Vapnik V. Writer-adaptation for on-line handwritten character recognition. In Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93), IEEE; 1993. pp. 187–191..

[27] Meng Z, Li J, Gong Y. Adversarial speaker adaptation. In: ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE; 2019. p. 5721–5.

[28] Platt JC, Matic NP. A constructive rbf network for writer adaptation. Advances in Neural Information Processing Systems 1997:765–71.

[29] Singh H, Sharma RK, Singh VP. Online handwriting recognition systems for indic and non-indic scripts: a review. Artif Intell Rev 2020:1–55.

[30] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15:1929–58.

[31] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In Advances in Neural Information Processing Systems; 2017..

[32] Wang ZR, Du J. Writer code based adaptation of deep neural network for offline handwritten chinese text recognition. In: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE. p. 548–53.

[33] Wang ZR, Du J, Wang JM. Writer-aware cnn for parsimonious hmm-based offline handwritten chinese text recognition. Pattern Recogn 2020;100:107102.

[34] Yang HM, Zhang XY, Yin F, Luo Z, Liu CL. Unsupervised adaptation of neural networks for chinese handwriting recognition. In: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR). p. 512–7. doi: https://doi.org/10.1109/ICFHR.2016.0100.

[35] Yang HM, Zhang XY, Yin F, Sun J, Liu CL. Deep transfer mapping for unsupervised writer adaptation. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). p. 151–6. doi: https://doi.org/10.1109/ICFHR-2018.2018.00035.

[36] Zhang XY, Bengio Y, Liu CL. Online and offline handwritten chinese character recognition: A comprehensive study and new benchmark. Pattern Recogn 2017;61:348–60.

[37] Zhang XY, Liu CL. Writer adaptation with style transfer mapping. IEEE Trans Pattern Anal Mach Intell 2013;35:1773–87. doi: https://doi.org/10.1109/TPAMI.2012.239.