CALText: Contextual Attention Localization for Offline Handwritten Text

CALText: Contextual Attention Localization for Offline Handwritten Text

Tayaba Anjum[1] and Nazar Khan[1*]

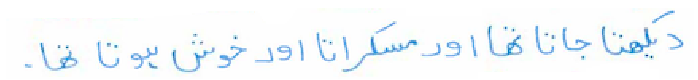
[1*]Department of Computer Science, University of the Punjab, Allama Iqbal Campus, Mall Road, Lahore, 54000, Punjab, Pakistan.

*Corresponding author(s). E-mail(s): nazarkhan@pucit.edu.pk; Contributing authors: tayaba.anjum@pucit.edu.pk;

Abstract

Recognition of Arabic-like scripts such as Persian and Urdu is more challenging than Latin-based scripts. This is due to the presence of a two-dimensional structure, context-dependent character shapes, spaces and overlaps, and placement of diacritics. We present an attention based encoder-decoder model that learns to read handwritten text in context. A novel localization penalty is introduced to encourage the model to attend only one location at a time when recognizing the next character. In addition, we comprehensively refine the only complete and publicly available handwritten Urdu dataset in terms of ground-truth annotations. We evaluate the model on both Urdu and Arabic datasets. For Urdu, contextual attention localization achieves **82.06%** character recognition rate and **51.97%** word recognition rate which represent more than **2×** improvement over existing bi-directional LSTM models. For Arabic, the model outperforms multi-directional LSTM models with **77.47%** character recognition rate and **37.66%** word recognition rate without performing any slant or skew correction. Code and pre-trained models for this work are available at https://github.com/nazar-khan/CALText.

Keywords: Handwritten Text Recognition, Cursive, Urdu, Arabic, Offline, Context, Attention, Localization, Encoder, Convolutional Neural Network, DenseNet, Decoder, Gated Recurrent Unit, Optical Character Recognition



دیکھتا جاتا تھا اور مسکراتا اور خوش ہوتا تھا

د ی ک ھ ت ا ج ا ت ا ت ھ ا ا و ر م س ک ر ا ت ا ا و ر خ و ش ھ و ت ا ت ھ ا

Fig. 1: **Top**: Handwritten Urdu text. **Middle**: Corresponding typed version. **Bottom**: Handwritten cursive ligatures appear significantly different from the corresponding isolated, typed characters. Arabic-like scripts such as Urdu and Persian are neither written nor typed in the form of isolated characters.

1 Introduction

Offline Handwritten Text Recognition (OHTR) refers to the problem of recognizing handwritten text from an image. OHTR provides the interface between humans and machines by automatic digitization of handwritten documents such as letters, lecture notes, bank checks, medical records, and library books. The OHTR problem varies from Optical Character Recognition (OCR) because of the wide variety of writing styles, sizes, sentence lengths, pen types, page backgrounds, and rule violations.

For Latin-based handwritten text, good quality datasets are readily available [33, 39, 40, 61]. Similar datasets exist for handwritten Arabic [2, 4, 6, 7, 37, 62] and Persian [53] as well. Recognition of Arabic-like scripts (Arabic, Persian and Urdu) presents interesting challenges in addition to those for Latin-based scripts [46]. For example, the presence of two-dimensional structure, character and ligature overlap, ligature deformation, context-sensitive shapes of ligatures, placement and quantities of diacritical marks, stretching, and spacing. Figure 1 depicts handwritten Urdu text in the first row, the corresponding typed text in the second row and the individual characters in the third row. It can be seen that within cursively written Urdu words, the individual characters appear with widely varying ligatures. While typed text conforms to some rules, handwriting can be restricted to very few rules. Despite being the 10th most spoken language in the world [1], there has been very little focus on the offline recognition of handwritten Urdu text. This dearth of methods and datasets for Urdu is reflected in other areas like natural language processing as well [12].

Previous attempts at OHTR for Arabic-like scripts have either used raw pixels or features extracted from a Convolutional Neural Network (CNN) that are then decoded using a recurrent model such as a bidirectional Long Short-Term Memory (BLSTM) network [4, 24] that aims to capture the horizontal nature of *deskewed* text. Multidimensional LSTM (MDLSTM) has also been employed to capture the horizontal as well as vertical nature of Arabic-like scripts [3]. These approaches use the complete image for recognition of a character at a certain time step. However, a single character can be recognized from a specific location of an image at a certain time step using attention [23].

Attention mechanism helps to recognize a character from an image by focusing on specific regions at a certain time step.

In previous efforts for Arabic-like scripts, error reporting has been restricted to the level of characters. World level recognition rates are not reported. This, we speculate, is due to the character level recognition rates not being good enough for recognition of words since a word is recognized correctly if and only if each and every character is recognized correctly.

Traditional variants of LSTM (1 direction), BLSTM (2 directions) and MDLSTM (4 directions) all limit the number of directions that a decoder can proceed in. In contrast, an attention-based decoder is direction-less since it can focus on any relevant part of the image at any given time. However, characters in a sequence can't appear at arbitrary locations. For text, *attention itself needs context*. That is, where to focus next depends on where the model has already focused previously. One such decoder was introduced by the authors in [8]. However, in solving problems related to recognition of a line of text, each character appears within a small contiguous region instead of being scattered all over the image. Therefore, in addition to the context of attention, text requires localization of attention as well. We show in this work how localization can be introduced into a contextual attention-based decoder. This results in our proposed *contextual attention localization* (CAL) method for handwritten text (CALText).

In this paper we present a thorough explanation of the encoder/decoder architecture from [8], extend it using attention localization and introduce a technique for compact spatiotemporal attention visualization. We use a DenseNet encoder to extract high level features from the input image. The features are then decoded into a contextual sequence of output characters using localized attention. Using localized attention in context encourages our model to attend only to relevant image regions while decoding an output sequence. The encoder and decoder are trained jointly so that the decoder can guide the encoder towards generation of richer, more useful features.

1.1 Contributions

The paper contributes as follows:

1. We present a detailed contextual attention-based, encoder-decoder model for the recognition of offline handwritten text. In particular, we describe our previous model [8] in greater detail and extend it.
2. We show that, armed with attention, direction-less decoders out-perform uni-directional, bidirectional and multi-directional decoders.
3. We show that for text recognition, encouraging the model to localize its attention improves recognition rates.
4. We comprehensively re-annotate the PUCIT-OHUL dataset [8] of offline handwritten Urdu text using 7 annotators.
    - Missing ground-truth lines are added.
    - We synchronize ground-truth text with the text on the images.

  - We record instances of text that is crossed-out, overwritten, or covered with correction fluid.
  - We annotate incorrectly spelled words as they are. We do not correct spelling mistakes in the written text.

5. Compared to previous approaches for handwritten Urdu recognition, we report 2× improvement of both character and word recognition rates.
6. We achieve state-of-the-art recognition rates on the Arabic KHATT dataset at both character and word level.
7. We introduce a method for compact visualization of spatiotemporal attention that helps explain the model's results.

For reproducible research, code, pretrained models, and links to datasets for this work are publicly available at https://github.com/nazar-khan/CALText.

1.2 Paper Organization

The rest of the paper is structured as follows. Section 2 reviews the state-of-the-art related to offline handwritten text recognition. Challenges specific to Urdu and other Arabic-like scripts are discussed in Section 3. This is followed by detailed explanations of our encoding and decoding frameworks in Section 4. Section 5 describes the datasets used for training and evaluation. Section 6 describes the experimental setup including network architectures, hyperparameter settings, and training details. Quantitative as well as qualitative experimental results are presented in Section 7 and the paper is concluded in Section 8.

2 Literature Review

Deep convolutional networks with some form of recurrence, attention or self-attention represent the state of the art for the OHTR problem for multiple scripts such as Latin-based [17, 36], Arabic [3], CJK[1] [35, 48, 65, 68] and even mathematical expressions [34] and music notes [11]. Historically, OHTR has evolved from convolutional [33] to recurrent [22] to attention-based [23] models.

Compared to Urdu, Arabic text recognition has a longer history. We refer the interested reader to surveys of traditional approaches and datasets in [27, 49] and more recent deep learning based approaches in [31]. Early efforts at Arabic and Urdu recognition were restricted to typed documents with fixed font styles and sizes [5, 28, 54]. Uni-, bi- and multi-directional LSTMs [43–45, 63, 64] were also applied for typed Urdu recognition which remains an active research area [42].

Handwritten text recognition is significantly more difficult compared to printed text as writing style, pen type, page background, and rule violations heavily impact the recognition process. Early attempts have naturally been restricted to recognition of isolated characters [14, 18, 21, 38, 47, 51] and

[1] Chinese Japanese Korean

character recognition continues to be studied for some languages [20, 25, 30, 59].

For Urdu, the CENPARMI-U dataset [56] focused on isolated characters and contained 44 handwritten Urdu characters and numerals, 5 special characters, and only 57 Urdu words. It also contained Urdu dates and numeral strings. Before the success of deep learning, support vector machines (SVMs) were the most commonly used classifiers for recognition of isolated handwritten Urdu numerals [13], characters [50] and words [41]. Most methods differed primarily in terms of which hand-designed features were used as input to the SVM.

Deep learning based solutions for offline handwritten Urdu text include [4] and [24], both of which employ bi-directional LSTM models. Until the year 2020, the only publicly available dataset usable for training deep learning models was UCOM/UNHD [4]. However, it was only partially available. The PUCIT-OHUL dataset was introduced by the authors in [8] and made publicly available. It has been refined in this paper.

A summary of existing offline text recognition techniques for Urdu and Arabic is presented in Table 1. Columns 5 and 6 provide a major reflection of the state of the art as most methods restrict error reporting to the level of characters only. Column 7 also exposes the lack of attention based modelling for Urdu and Arabic.

Table 1: Summary of existing offline text recognition techniques for Urdu and Arabic. Last 5 columns should be interpreted as follows. **CRR**: results reported in terms of character recognition rate, **WRR**: in terms of word recognition rate, **Rec.**: uses recurrent decoder, **Att.**: uses decoder with attention, **SF.**: segmentation free method.

Typed Text Lines							
Method	**Dataset**	**Lang**	**CRR**	**WRR**	**Rec.**	**Att.**	**SF**
CNN-BLSTM [63]	UPTI	Urdu	✓	×	✓	×	✓
Raw-LSTM [64]	UPTI	Urdu	×	×	✓	×	✓
Zoning Features + BLSTM [43]	UPTI	Urdu	✓	×	✓	×	✓
Raw-MDLSTM [44]	UPTI	Urdu	✓	×	✓	×	✓
CNN-MDLSTM [45]	UPTI	Urdu	✓	×	✓	×	✓
CNN-LSTM [42]	MMU-OCR21	Urdu	✓	✓	✓	×	✓
Handwritten Characters/Words							
Method	**Dataset**	**Lang**	**CRR**	**WRR**	**Rec.**	**Att.**	**SF**
Run-length + SVM [13]	Custom	Urdu	×	-	×	×	×
Gradient features + SVM [55]	CENPERMI-U	Urdu	×	-	×	×	×
Handwritten Text Lines							
Method	**Dataset**	**Lang**	**CRR**	**WRR**	**Rec.**	**Att.**	**SF**
Raw-LSTM [4]	UCOM/UNHD	Urdu	✓	×	✓	×	✓
CNN-BLSTM [24]	Custom	Urdu	✓	×	✓	×	✓
Raw-MDLSTM [3]	KHATT	Arabic	✓	×	✓	×	✓
CNN-BLSTM + RL [23]	KHATT	Arabic	✓	×[a]	✓	✓	✓
Proposed	KHATT PUCIT-OHUL	Arabic Urdu	✓	✓	✓	✓	✓

[a]WRR reported in [23] is actually percentage of correctly recognized words. It is not an edit-distance based metric.

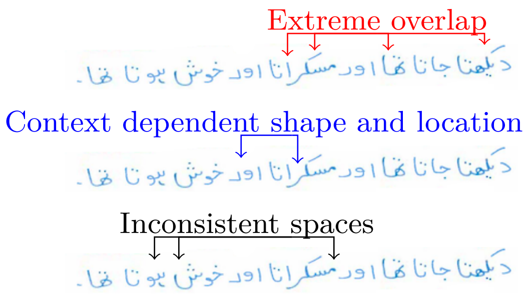


Fig. 2: Challenges in offline handwritten Arabic-like scripts. This Urdu sentence contains i) extreme overlap between ligatures due to the 2-dimensional structure of Urdu, ii) context dependent ligature shapes as well as locations, and iii) intra-word spaces occasionally larger than inter-word spaces.

3 Challenges of OHTR

Recognition of Arabic-like scripts such as Urdu is difficult due to Sayre's paradox [57] that a cursively written word cannot be recognized without being segmented and cannot be segmented without being recognized. In addition, handwritten text can employ a wide variety of pen types, writing styles, sizes and page backgrounds. But most importantly, in handwritten text, almost every rule can be violated until the text becomes illegible. Some of the main challenges in offline handwritten text recognition of Arabic-like scripts, as explained in Figure 2, are as follows.

1. Cursive Arabic-like scripts can have extreme overlap among characters. Sometimes, parts of a letter can overlap multiple other letters or even other words. In other cases, letters can lie exactly on top of other letters.
2. Shape and location of letters is context-dependent. The placement of characters is not just in right-to-left order. Arabic-like scripts have a 2-dimensional structure as well.
3. Arabic-like scripts are not entirely cursive. They are a mix of cursive and non-cursive. As a result, sometimes the space within characters of a word can be larger than the space between words. This makes tokenization of a sentence into words difficult.
4. Within cursive words, characters to be recognized often appear with widely varying ligatures. While typed text conforms to some rules, handwriting can be restricted to very few rules. As long as the text is legible, almost any rule can be violated by the writer.

Therefore, handwritten Urdu and Arabic recognition is not a trivial problem.

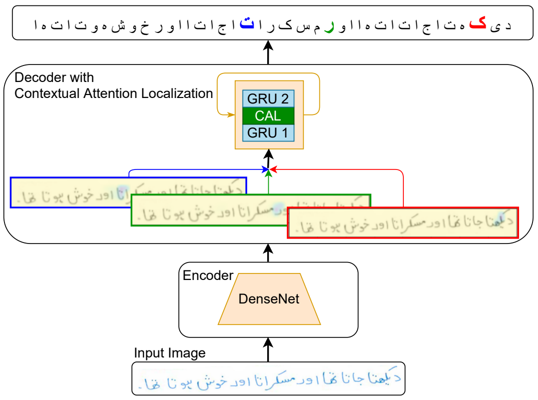


Fig. 3: Proposed encoder/decoder framework. An image containing handwritten text is input to a DenseNet encoder. Extracted features are passed through a decoder that attends to specific image regions in context so as to produce probabilities of output characters at every time step. The probabilities are finally decoded into output characters using beam search.

4 Methodology

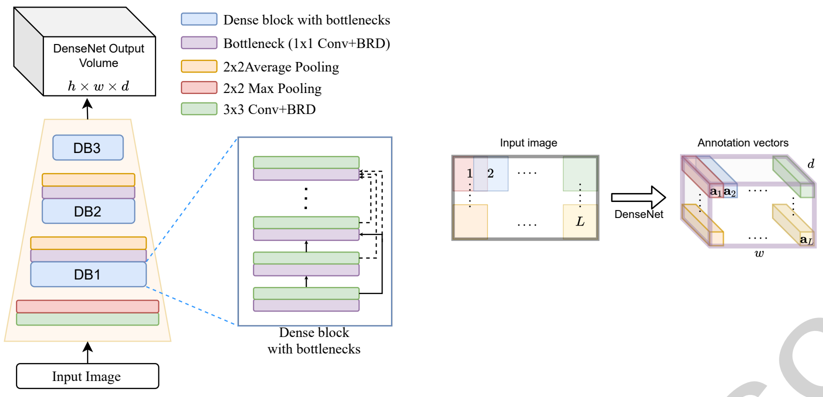
For handwritten text, even if the input images are restricted to have a fixed size, the characters may constitute a variable-length sequence. The encoder/decoder model with a recurrent decoder can be used for the recognition of such variable text sequences [15]. The encoder transforms an input image into an intermediate representation. The decoder then transforms the intermediate representation into a sequence of output characters. While decoding, an attention mechanism is used to focus on a specific part of the image to produce different characters. An overview of our approach is shown in Figure 3. Given an input image, the model produces a variable-length sequence of 1-hot vectors

$$Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\} \tag{1}$$

where $\mathbf{y}_t \in \mathbb{R}^k$ is the 1-of-k encoding of the t-th character in the image text containing T characters including any blank spaces and punctuation marks. Size of the vocabulary is denoted by k.

4.1 Encoder: DenseNet with Bottlenecks

For extraction of high-level salient features, we use the DenseNet [26] architecture which is a variation of CNNs with dense blocks. In a dense block, convolution is applied to all previous layers within the block instead of just the previous layer. The use of dense blocks provides a more diverse set of features by minimizing information loss during forward and backward propagation. Since connecting all previous layers becomes costly, bottleneck layers (1×1 convolutions) are inserted to compress the volume of previous layers.



(a) (b)

Fig. 4: (a) Architecture of the DenseNet encoder that transforms an input image into a 3-dimensional feature volume. BRD = Batchnorm + ReLU + Dropout. (b) DenseNet output volume of size $h \times w \times d$ can be interpreted as d-dimensional *annotation vectors* of $h \times w$ overlapping blocks of the input image.

The convolutional output volume of layer i is calculated as

$$\mathbf{F}_i = \mathcal{C}_i(\mathcal{B}_i([\mathbf{F}_1;\ \mathbf{F}_2;\ \ldots;\ \mathbf{F}_{i-1}])) \tag{2}$$

where $\mathcal{C}_i(\cdot)$ represents regular convolution layers, $\mathcal{B}_i(\cdot)$ represents 1×1 convolution of a bottleneck layer and $[\mathbf{F}_1;\ \mathbf{F}_2;\ \ldots;\ \mathbf{F}_{i-1}]$ represents the depth-wise concatenation of the outputs of all previous layers. Pooling layers between dense blocks are used for extraction of features at multiple scales. The detailed architecture of our DenseNet encoder is shown in Figure 4a. There are 3 dense blocks. Each block consists of 32 sub-layers alternating between 1×1 convolution bottleneck layers and 3×3 convolution layers. Each dense block is preceded by a 2×2 pooling layer with stride 2.

The output of the dense encoder, for a given input image, is a volume of dimensions $h \times w \times d$. It can be interpreted as d-dimensional *annotation vectors* of $h \times w$ overlapping blocks of the input image. This interpretation is illustrated in Figure 4b. The correspondence between the annotation vectors and image regions is because convolution represents moving, overlapping, and localized dot-products and pooling results in effectively larger receptive fields. Therefore each depth vector in the output volume corresponds to a d-dimensional encoding of a localized image region. The set of all annotation vectors produced by the DenseNet is represented by

$$A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_L\} \tag{3}$$

where $\mathbf{a}_p \in \mathbb{R}^d$ and number of regions is given by $L = hw$.

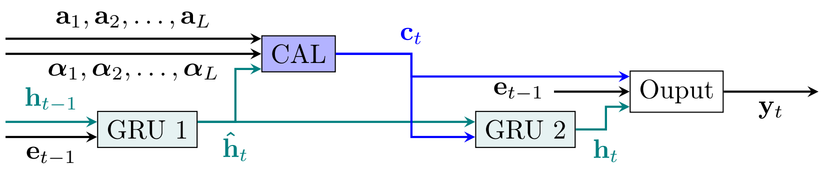


Fig. 5: The decoder contains a contextual attention localization (CAL) unit between two gated recurrent units (GRUs). This diagram represents the procedures from Equation (5) through Equation (18). Vector $\boldsymbol{\alpha}_p = [\alpha_{1p}, \alpha_{2p}, \ldots, \alpha_{(t-1)p}]$ contains the history of attention values for location p until time $t-1$.

4.2 Decoder: Gated Recurrent Units with Contextual Attention Localization

When an encoder extracts high-level visual features from input images, a Gated Recurrent Unit (GRU) [16] can be used to generate a variable-length text sequence character by character, conditioned on the previous output character $\mathbf{y}_{t-1}$, the current hidden state $\mathbf{h}_t$ and a context vector $\mathbf{c}_t$. Our decoder consists of a Contextual Attention Localization (CAL) unit sandwiched between two GRUs as shown in Figure 5. This first operation of the decoder is to obtain a dense, low-dimensional embedding $\mathbf{e}_{t-1}$ of the last output character $\mathbf{y}_{t-1}$ through the projection

$$\mathbf{e}_{t-1} = \mathbf{E}\mathbf{y}_{t-1} \tag{4}$$

where $\mathbf{E} \in \mathbb{R}^{m \times k}$ is a lower-dimensional projection matrix for getting rid of the wasteful 1-of-k coding of character vector $\mathbf{y}_{t-1}$. Matrix $\mathbf{E}$ is learned as part of the network parameters.

4.2.1 First GRU

The first GRU is used to compute a potential hidden state $\hat{\mathbf{h}}_t$ based on previous hidden state $\mathbf{h}_{t-1}$ and previous output character embedding $\mathbf{e}_{t-1}$ as explained in Figure 6. The hidden state is computed through soft-gating mechanisms within the GRU at each time step. The GRU contains

1. an update gate $\mathbf{z}_{1,t}$ that regulates the role of past and present in updating the hidden states, and
2. a reset gate $\mathbf{r}_{1,t}$ that determines how much of the past is to be retained.

Both gates are parameterized as linear transformations followed by sigmoidal non-linearities.

Computation of update gate $\mathbf{z}_{1,t}$ Word embedding $\mathbf{e}_{t-1}$ and recurrent state $\mathbf{h}_{t-1}$ are projected into an l-dimensional latent space via learnable matrices $\mathbf{W}_{ez} \in \mathbb{R}^{l \times m}$ and $\mathbf{U}_{hz} \in \mathbb{R}^{l \times l}$ respectively. The latent space projections are added to a learnable bias vector $\mathbf{b}_{z_1} \in \mathbb{R}^l$ and passed through a logistic sigmoid function to obtain the update gate

$$\mathbf{z}_{1,t} = \sigma(\mathbf{W}_{ez}\mathbf{e}_{t-1} + \mathbf{U}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_{z_1}) \tag{5}$$

Computation of reset gate $\mathbf{r}_{1,t}$ Computation of the reset gate is identical to the update gate except that it has its own latent space with a pair of learnable projection matrices $\mathbf{W}_{er} \in \mathbb{R}^{l\times m}$ and $\mathbf{U}_{hr} \in \mathbb{R}^{l\times h}$ and bias vector $\mathbf{b}_{r_1} \in \mathbb{R}^{l}$. The reset gate is computed as

$$\mathbf{r}_{1,t} = \sigma(\mathbf{W}_{er}\mathbf{e}_{t-1} + \mathbf{U}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_{r_1}) \tag{6}$$

Computation of potential hidden state $\widetilde{\mathbf{h}}_t$ A new potential hidden state can be computed as another combination of latent space projections of embedding $\mathbf{e}_{1,t}$ and previous state $\mathbf{h}_{t-1}$.

$$\widetilde{\mathbf{h}}_t = \tanh(\mathbf{W}_{eh}\mathbf{e}_{t-1} + \mathbf{r}_{1,t} \otimes (\mathbf{U}_{rh}\mathbf{h}_{t-1}) + \mathbf{b}_{h_1}) \tag{7}$$

where $\mathbf{W}_{eh} \in \mathbb{R}^{l\times m}$ and $\mathbf{U}_{rh} \in \mathbb{R}^{l\times l}$ are learnable projection matrices and $\mathbf{b}_{h_1} \in \mathbb{R}^{l}$ is a learnable bias vector. The role of the reset gate $\mathbf{r}_{1,t}$ can be seen here as regulating retention of the past $\mathbf{h}_{t-1}$ through element-wise multiplication. The tanh() non-linearity is to ensure that the potential state values lie in the range $(-1, 1)$.
Computation of predicted hidden state $\hat{\mathbf{h}}_t$With both gates $\mathbf{z}_{1,t}$ and $\mathbf{r}_{1,t}$ and a potential hidden state $\widetilde{\mathbf{h}}_t$ computed, the predicted hidden state of the first GRU can finally be updated through linear combination of previous and potential states as

$$\hat{\mathbf{h}}_t = \mathbf{z}_{1,t} \otimes \mathbf{h}_{t-1} + (1 - \mathbf{z}_{1,t}) \otimes \widetilde{\mathbf{h}}_t \tag{8}$$

where the role of the update gate $\mathbf{z}_{1,t}$ can be seen as regulating retention of the past $\mathbf{h}_{t-1}$ and addition of the present $\widetilde{\mathbf{h}}_t$ through element-wise multiplications.

We next describe how the predicted hidden state $\hat{\mathbf{h}}_t$ is then used in the dynamic computation of the attention weights.

4.2.2 Contextual Attention (CA)

Attention has been successfully used in computer vision [10] and also in handwriting recognition [70, 71]. The benefit of using attention is that it adds context to the decoder. For example, an image region should be classified as a dog by looking at the dog and not by looking at its owner. While a dog can appear anywhere in an image, characters in a text sequence cannot appear at arbitrary locations. Text has a strong structure and therefore *attention itself needs context*. In simpler words, where to focus next depends on where the model has focused previously.

Ideally, we would like our text recognizer to *read* like we do. That is, it should focus on the relevant image region when recognizing each character or

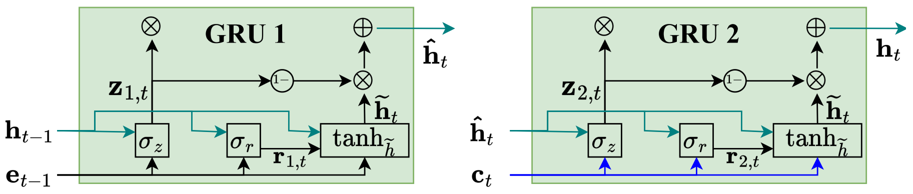


Fig. 6: Internal structure of GRUs used in the decoder. Each □ represents a learnable vector transformation followed by a non-linearity and each ◯ represents an element-wise operation. **Left**: The first GRU outputs predicted hidden state $\hat{\mathbf{h}}_t$ based on previous hidden state $\mathbf{h}_{t-1}$ and embedding $\mathbf{e}_{t-1}$ of previous output word. **Right**: The second GRU takes predicted hidden state $\hat{\mathbf{h}}_t$ and attention context vector $\mathbf{c}_t$ and computes a final hidden state $\mathbf{h}_t$ that is now dependent on the history of attentions.

word. However, for images containing text, multiple instances of a character or word can appear at multiple locations. Nothing stops an attention model from re-attending a previously attended location or from giving the right answer by attending the wrong location. Therefore, *for text*, the decision for attention needs to depend on the history of attention. In our model, attention at time t is made to depend on previous values of attention. This is achieved through a coverage vector, that represents a history of attention already given to each location.

Let attention weight α_{tl} denote the importance of image patch p at time t. By aggregating the attentions through time, we can compute the $h \times w$ aggregated attention array S^{α} at time t via

$$S^{\alpha}_{tp} = \sum_{\tau=1}^{t-1} \alpha_{\tau p} \tag{9}$$

The 2-dimensional array S^{α} can be convolved with q filters of size $f \times f$ arranged as a volume Q to obtain an $h \times w \times q$ volume F_t as

$$F_t = S^{\alpha} * Q \tag{10}$$

Features in F_t can be interpreted as the *history of attention* until time t and Q is a learnable layer of convolution filters. Similar to how annotation vector $\mathbf{a}_p$ is formed, the coverage vector $\mathbf{f}_p$ is the q-dimensional vector at location p of volume F_t as shown in Figure 7a. Each vector $\mathbf{f}_p$ can be interpreted as the *coverage* given to location p until time t. Therefore, $\mathbf{f}_p$ is called a *coverage vector*. The coverage vectors $\mathbf{f}_1, \ldots, \mathbf{f}_L$ describe the attention given *so far* to different regions in a *dynamic* fashion. In contrast, the annotation vectors $\mathbf{a}_1, \ldots, \mathbf{a}_L$ describe input image regions in a *static* fashion.

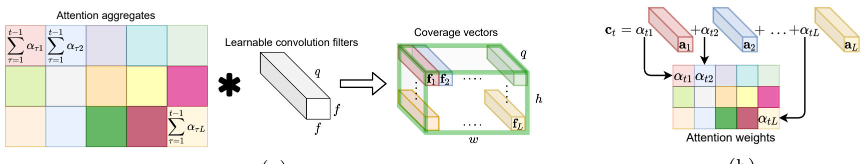

Attention aggregates
Learnable convolution filters
Coverage vectors
$\mathbf{c}_t = \alpha_{t1}\mathbf{a}_1 + \alpha_{t2}\mathbf{a}_2 + \ldots + \alpha_{tL}\mathbf{a}_L$
Attention weights
(a)
(b)

Fig. 7: (a) Coverage vectors $\mathbf{f}_i \in \mathbb{R}^q$ corresponding to each image region are dynamically computed using the history of attention up till the previous time step $t-1$. (b) Attention weight α_{tp} determines importance of location p in determining context vector $\mathbf{c}_t$ at time t.

The dynamic attention values α_{tp} for the current time step t can then be computed via

$$e_{tp} = \mathbf{v}_a^T \tanh(\mathbf{W}_a \hat{\mathbf{h}}_t + \mathbf{U}_a \mathbf{a}_p + \mathbf{U}_f \mathbf{f}_p + \mathbf{b}_a) \tag{11}$$

$$\alpha_{tp} = \frac{\exp(e_{tp})}{\sum_{u=1}^{L} \exp(e_{tu})} \tag{12}$$

where $\mathbf{v}_a \in \mathbb{R}^n$, $\mathbf{W}_a \in \mathbb{R}^{n\times l}$, $\mathbf{U}_a \in \mathbb{R}^{n\times d}$, and $\mathbf{U}_f \in \mathbb{R}^{n\times q}$ are all learnable projection parameters and $\mathbf{b}_a \in \mathbb{R}^n$ is a learnable bias vector. Notice that the softmax operation in Equation (12) ensures that attention weights α_{tp} at time t represent probabilities.

A *dynamic* representation of the relevant part of the input image at time t can now be defined. As illustrated in Figure 7b, it is computed as the expectation of the annotation vectors

$$\mathbf{c}_t = \sum_{p=1}^{L} \alpha_{tp} \mathbf{a}_p \tag{13}$$

where attention weights α_{tp} determine the importance of each image region at time t. They determine the image *context* in which the decision for output character at time t will be made. Accordingly, the expectation of the annotation vectors can be termed as the *context vector* $\mathbf{c}_t$.

4.2.3 Second GRU

The second GRU computes the final hidden state $\mathbf{h}_t$ using predicted hidden state $\hat{\mathbf{h}}_t$ and the context vector $\mathbf{c}_t$ as explained in Figure 6. The computation of update gate $\mathbf{z}_{2,t}$, reset gate $\mathbf{r}_{2,t}$, and candidate hidden state $\widetilde{\mathbf{h}}_t$ is identical to the computation performed in the first GRU. However, the input of the second GRU is the context vector $\mathbf{c}_t$ and the predicted hidden state $\hat{\mathbf{h}}_t$ from the first GRU. The final hidden state $\mathbf{h}_t$ of the decoder is computed as

$$\mathbf{z}_{2,t} = \sigma(\mathbf{C}_{cz}\mathbf{c}_t + \mathbf{U}'_{hz}\hat{\mathbf{h}}_t + \mathbf{b}_{z_2}) \tag{14}$$

$$\mathbf{r}_{2,t} = \sigma(\mathbf{C}_{cr}\mathbf{c}_t + \mathbf{U}'_{hr}\hat{\mathbf{h}}_t + \mathbf{b}_{r_2}) \tag{15}$$

$$\widetilde{\mathbf{h}}_t = \tanh(\mathbf{C}_{ch}\mathbf{c}_t + \mathbf{r}_{2,t} \otimes (\mathbf{U}'_{rh}\hat{\mathbf{h}}_t) + \mathbf{b}_{h_2}) \tag{16}$$

$$\mathbf{h}_t = \mathbf{z}_{2,t} \otimes \hat{\mathbf{h}}_t + (1 - \mathbf{z}_{2,t}) \otimes \widetilde{\mathbf{h}}_t \tag{17}$$

where learnable projection matrices can be divided into context transformations $\{\mathbf{C}_{cz}, \mathbf{C}_{cr}, \mathbf{C}_{ch}\} \in \mathbb{R}^{l\times d}$ and recurrent transformations $\mathbf{U}'_{hz}, \mathbf{U}'_{hr}, \mathbf{U}'_{rh}\} \in \mathbb{R}^{l\times l}$. The learnable bias vectors $\{\mathbf{b}_{z_2}, \mathbf{b}_{r_2}, \mathbf{b}_{h_2}\} \in \mathbb{R}^l$ are specific to the second GRU.

4.2.4 Output

After obtaining the output of the second GRU, the previous character embedding $\mathbf{e}_{t-1}$, hidden state $\mathbf{h}_t$, and context vector $\mathbf{c}_t$ are passed through

1. an affine layer,
2. followed by a 1-dimensional max-pooling layer with stride 2,
3. followed by a dropout layer,
4. followed by another affine layer,
5. followed by a softmax layer

to compute conditional probabilities of the next character. The computation of all three layers can be compactly summarized as

$$p(\mathbf{y}_t \mid \mathbf{y}_{t-1};\ I) = \mathcal{S}\left(\mathbf{W}_o\mathcal{D}\left(\mathcal{M}\left(\mathbf{W}_e\mathbf{e}_{t-1} + \mathbf{W}_h\mathbf{h}_t + \mathbf{W}_c\mathbf{c}_t + \mathbf{b}_w\right)\right) + \mathbf{b}_o\right) \tag{18}$$

where I is the input image, $\mathcal{S}(\cdot)$ is the softmax activation function, $\mathcal{M}(\cdot)$ is a 1-dimensional max-pooling layer with stride 2, $\mathcal{D}(\cdot)$ is a dropout layer, $\mathbf{W}_o \in \mathbb{R}^{k\times m}$, $\mathbf{W}_e \in \mathbb{R}^{m\times m}$, $\mathbf{W}_h \in \mathbb{R}^{m\times l}$ and $\mathbf{W}_c \in \mathbb{R}^{m\times d}$ are all learnable projection matrices, and $\mathbf{b}_w \in \mathbb{R}^m$ and $\mathbf{b}_o \in \mathbb{R}^k$ are learnable bias vectors. The dropout layer is only used during training.
Training For training, we employ teacher forcing [66] whereby the previous ground-truth character (instead of the previous output character or probability vector) is fed as decoder input for the current time step. This helps speed up training by i) nullifying the cumulative effect of mistakes made in earlier decoding, and ii) avoiding large decoding search spaces associated with higher likelihood decoding methods such as beam search [67].
Testing For inference, the sequence of probability vectors output by the decoder is converted into an optimal sequence of output characters $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T$ via beam search whereby only the b most-probable branches are pursued at each time step.

4.3 Contextual Attention Localization (CAL)

Text is made up of very localized image regions so that a single character appears in a small contiguous area. It cannot appear scattered all over the image. Therefore, in addition to context of attention, text requires *localization* *of attention* as well. This behavior can be induced by encouraging the attention weights α_{tl} to be sparse. Accordingly, we define the attention localization penalty as

$$\mathcal{L}^{\mathrm{L}}(\theta) = \sum_{t=1}^{T}\sum_{l=1}^{L} |\alpha_{tl}| \tag{19}$$

which is the ℓ_1-norm of the vector of attention values. Minimizing ℓ_1-norm encourages only few α_{tl} to be large and all the rest to be close to zero. This discourages the model from attending to multiple locations at the same time.

4.4 Loss Function

The proposed encoder-decoder model is trained jointly for recognition of offline handwritten text. In the model, the next character is predicted depending on the previous character and input image as explained in Equation (18). The objective of the training is to maximize the probability of the correct character at each time step. In order to train the model to classify using *contextual attention*, regularized cross-entropy is used as the loss function

$$\mathcal{L}_n^{\mathrm{CA}}(\theta) = -\sum_{t=1}^{T_n} \ln\left(p(\mathbf{y}_{nt} \mid \mathbf{t}_{nt}, I_n)\right) + \lambda r(\theta) \tag{20}$$

where T_n is the number of characters in the ground-truth of the n^{th} training image I_n, and $\mathbf{t}_{nt}$ and $\mathbf{y}_{nt}$ are 1-hot vectors of the target and predicted characters, respectively, at time step t. Function $r(\theta)$ represents the ℓ_2-norm of all weights except those from the convolution layers and $\lambda > 0$ is the regularization hyperparameter. A model trained by minimizing loss function (20) corresponds to the method proposed in [8].

In order to train the model to classify using the proposed *localized contextual attention*, regularized cross-entropy (20) can be augmented with our attention localization penalty (19) to yield our proposed loss function

$$\mathcal{L}_n^{\mathrm{CAL}}(\theta) = \mathcal{L}_n^{\mathrm{CA}}(\theta) + \gamma\mathcal{L}_n^{\mathrm{L}}(\theta) \tag{21}$$

where hyperparameter $\gamma > 0$ controls the tradeoff between classification and localization.

5 Datasets

We evaluate our proposed method on handwritten Urdu as well as Arabic.

5.1 Urdu Dataset

The PUCIT-OHUL dataset introduced in [8] is a multi-writing style dataset with different pen types, ink types, text sizes, and background types and colors. It was collected using 100 students between 20 and 24 years of age. A total of

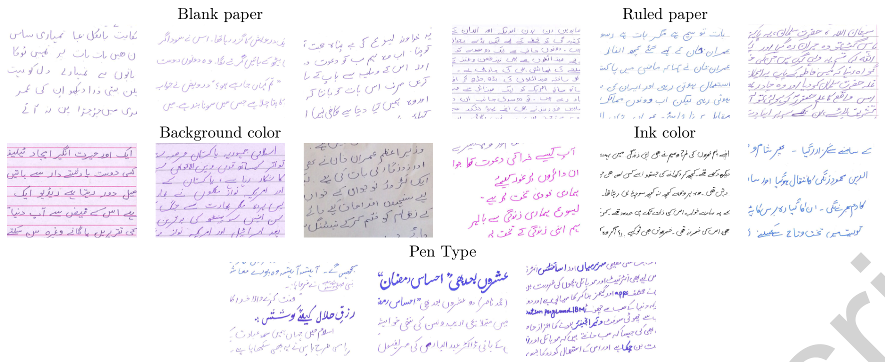


Fig. 8: Sample page images from the PUCIT-OHUL dataset demonstrating challenges related to different paper types, backgrounds, colors, ink types and pen types.

479 pages of text were collected. Each page was scanned at 200 DPI and text lines were manually segmented. The line images were not deskewed. As shown in Figure 8, the dataset contains different types of page backgrounds, page colors, ink types and pen types. An overview of the subject categories covered in the dataset is shown in Table 2. The writers were not given any instructions regarding what subject to choose and were completely free in their choices. The detailed vocabulary of unique characters and symbols extracted from the dataset is given in Table 3.

Unsurprisingly, labeling errors in datasets adversely impact training as well as error rates [9]. Accordingly, we have re-annotated the complete PUCIT-OHUL dataset. The key outcomes of our re-annotation process are as follows.

1. We have added 92 new lines to the original test set.
2. In an attempt to reduce annotator bias, 7 annotators were used.
3. Vocabulary size was increased from 98 in the original ground-truth to 130 in our updated version. We annotated 31 additional characters from the English alphabet and 1 from Urdu that had been used by the original writers but the original ground-truth did not include them. In the updated annotations, we label each and every character.
4. For a few cases, there was a mismatch between images and annotations. Such mismatches have been fixed.
5. In the original ground-truth, spelling mistakes in the handwritten text were sometimes *corrected* by the annotators. In our re-annotations, such mistakes have been recorded as they appear with no attempt to correct them.
6. Original ground-truth had a non-trivial amount of errors in the annotation of diacritics. Since diacritics are usually small and the mind can *auto-correct* mistakes in diacritics, such errors are easy to overlook. A lot of attention was focused on correct re-annotation of diacritics.

7. We categorize crossed-out text as either readable or non-readable. Such information can be used to *not penalize* a recognizer that correctly recognizes crossed-out *but still readable* text.

Table 4 compares the PUCIT-OHUL dataset with existing datasets. Existing datasets are problematic for two reasons. Firstly, sometimes data statistics are not reported correctly. For instance, UCOM/UNHD [4] claims to have

Table 2: Categorization of the content in the PUCIT-OHUL dataset in terms of subjects, references, pages, and lines.

	Subject	**References**	**Pages**	**Lines**
1	Art	1	5	72
2	World	11	52	805
3	Economy	3	15	277
4	Social	19	84	1291
5	Literature	17	83	1293
6	Wildlife	1	5	106
7	Sports	4	21	317
8	Health	1	5	77
9	Religion	8	40	662
10	Nature	2	9	129
11	Science	3	14	203
12	Politics	14	67	1005
13	History	10	52	683
14	Education	6	27	479
	Total	100	479	7,399

Table 3: Vocabulary extracted from the PUCIT-OHUL dataset.

Category Name	**Symbols**														
Urdu Letters (40)	ا	آ	ب	پ	ت	ٹ	ث	ج	چ	ح	خ	د	ڈ	ذ	ر
	ڑ	ز	ژ	س	ش	ص	ض	ط	ظ	ع	غ	ف	ق	ک	گ
	ل	م	ن	ں	و	ہ	ھ	ء	ی	ے					
English Letters (31)	A	B	E	F	I	K	M	P	R	S	Y	a	b	c	d
	e	f	g	i	l	m	n	o	p	r	s	t	u	v	x
	y														
Counting (20)	0	1	2	3	4	5	6	7	8	9	۰	۱	۲	۳	۴
	۵	٦	۷	۸	۹										
Diacritics (24)	؛	ٓ	ً	َ		ٰ	ٗ	ؐ	ؑ	ؒ	ؒ	ؔ	ؓ	,	،
		"	'	:	ٍ	“	”	,	^						
Special Symbols (15)	[	]	/	(	)	؟	؎	...	.	؛	!	;	-	space	EOL

Table 4: Comparison with existing offline handwritten Urdu text datasets

Dataset	**Total Lines**	**Total Words**	**Total characters**	**Total Writers**	**Vocabulary Size**
UCOM/UNHD [4]	10,000	312,000	1,872,000	500	59
CENIP-UCCP [52]	2,051	23,833	-	200	-
Custom dataset [24]	6,000	86,400	432,000	600	-
PUCIT-OHUL	**7,401**	**80,059**	**367,925**	**100**	**130**

10,000 text lines, but only 700 of those lines are unique in terms of semantic content. Secondly, all 3 datasets in [4, 24, 52] are either publicly unavailable or only partially available. In contrast, the PUCIT-OHUL dataset with our updated annotations is publicly available[2] in its entirety.

The dataset is divided randomly into training, validation, and testing sets with respect to writers. Out of the 100 writers, we randomly selected 75 writers for the training set, 13 for the validation set and the remaining 12 for the testing set. In terms of lines, 5580 (75.42%) of the lines were included in the training set, 907 (12.33%) in the validation set, and 912 (12.25%) in the testing set.

5.2 Arabic Dataset

The KFUPM Handwritten Arabic TexT (KHATT) [37] dataset is an unconstrained offline handwritten Arabic dataset. We use the default split of the unique text portion of the dataset which consists of 4,825 training, 937 validation, and 966 testing line images. The dataset contains 63 unique symbols.

6 Experimental Setup

6.1 Experimental Datasets and Image Preprocessing

Three benchmark datasets PUCIT-OHUL, KHATT, and IAM datasets are used to evaluate the performance of CALText model. In each dataset, all training, validation and testing images are resized to 100×800 pixels. Before resizing, if the width is less than 300, we first double the width of the image by extra white padding and then resize the result to 100×800. This way we avoid excessive stretching in images with small width. To reduce overfitting, 4% salt and pepper noise is added to training images with probability of salt being 20% and probability of pepper being 80%. The gray values of all the images are normalized in range $[0, 1]$.

6.2 Network Architectures

To compare performance of proposed model with other state-of-the-art techniques used for offline handwritten text line recognition, we design/use multiple models as listed below.

1. CNN-GRU
2. CNN-LSTM
3. CNN-BGRU
4. CNN-BLSTM [24]
5. SimpleHTR [58][3]
6. LineHTR [32][4]

[2]http://faculty.pucit.edu.pk/nazarkhan/work/urdu_ohtr/pucit_ohul_dataset.html
[3]https://github.com/githubharald/SimpleHTR
[4]https://github.com/lamhoangtung/LineHTR

7. CRNN [60]
8. CNN-Contextual Attention
9. DenseNet-Contextual Attention
10. DenseNet-CALText

Table 5 describes the CNN used as the encoder in models 1-4 and model 8. In models 1-4, the feature volume produced by the last max-pooling operation has size $\lceil \frac{H}{32} \rceil \times \lceil \frac{W}{8} \rceil \times 512$. The reshape operation converts this 3-dimensional volume to a 2-dimensional representation of size $\lceil \frac{W}{8} \rceil \times 512 \lceil \frac{H}{32} \rceil$. The last dense layer then produces an output of size $\lceil \frac{W}{8} \rceil \times 64$ representing 64-dimensional encodings of horizontally scanned image regions. For model 8, since decoding with contextual attention requires a 3-dimensional feature volume as input, we discard the final dense layer and the last two max-pooling layers to obtain an encoded volume of size $\lceil \frac{H}{8} \rceil \times \lceil \frac{W}{8} \rceil \times 512$. Table 5 also describes the DenseNet encoder used in our models (9 and 10). In the DenseNet, the constant growth_rate is set to 24. It ensures equal number of output channels from each dense block. The compression ratio of 0.5 is used to check the growth of concatenated channels when using dense skip connections. Details of all decoders used in the experiments are provided in Table 6.

6.2.1 Hyperparameters Setting

Proposed Model The proposed model is composed of a DenseNet encoder and a contextual attention mechanism sandwiched between two GRU decoders. The network hyperparameters include dimensions of all learnable matrices, dropout ratio, regularization parameters, and optimization algorithm. Dimensions of all learnable matrices and vectors in our decoding framework are given in Table 7. During training, we use batch normalization and dropout. Dropout is only used at convolution layers with 0.2 drop ratio. We also apply regularization with $\lambda = 1e-4$ in the cross-entropy loss (20) and $\gamma = 1$ in the overall error function (21). Both values were fixed after cross-validation. We use an ADADELTA optimizer [69] with gradient clipping for better optimization. Training is continued for a maximum of 50 epochs until convergence of the training loss and the model with lowest validation loss is selected for testing.
Compared Models For comparative results, other state-of-the-art models (CNN-GRU, CNN-LSTM, CNN-BGRU, CNN-BLSTM, SimpleHTR, Line-HTR, and CRNN) are also trained on the PUCIT-OHUL and KHATT datasets. The models are trained until convergence of the training loss for a maximum of 50 epochs and the model with lowest validation loss is selected for testing. We used batch size 20, learning rate $1e-3$, dropout ratio 0.2 and the optimization algorithm was ADAM [29].

6.2.2 Experimental Platform

For training purposes, we used TensorFlow and Keras on an NVIDIA K80 GPU with 12GB RAM. The training curves of all models on the PUCIT-OHUL dataset are shown in Figure 9.

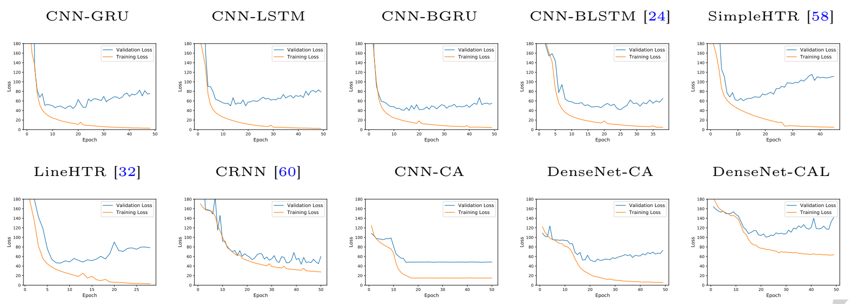


Fig. 9: Training curves of all models on the PUCIT-OHUL dataset.

Table 5: Architectural details of the CNN and DenseNet encoders used in the experiments.

CNN	**DenseNet**
Input size: $H \times W \times C$	Input size: $H \times W \times C$
3×3 Conv, 64 Batch norm, ReLU Dropout(0.2) Max pool, 2×2	7×7 Conv, 48, Stride 2 Batch norm, ReLU Max pool, 2×2
3×3 Conv, 128 Batch norm, ReLU Dropout(0.2) Max pool, 2×2	Dense block 1, 16 layers $\left\{\begin{array}{l} 1 \times 1 \text{ Conv}, 4 \times \text{growth\_rate} \\ \text{Batch norm, ReLU} \\ \text{Dropout}(0.2) \\ 3 \times 3 \text{ Conv, growth\_rate} \\ \text{Batch norm, ReLU} \\ \text{Dropout}(0.2) \end{array}\right.$ 1×1, Conv, Compression ratio 0.5 Avg pool, 2×2
3×3 Conv, 256 Batch norm, ReLU 3×3, Conv, 256 Batch norm, ReLU Dropout(0.2) Max pool, 2×2	Dense block 2, 16 layers $\left\{\begin{array}{l} 1 \times 1 \text{ Conv}, 4 \times \text{growth\_rate} \\ \text{Batch norm, ReLU} \\ \text{Dropout}(0.2) \\ 3 \times 3 \text{ Conv, growth\_rate} \\ \text{Batch norm, ReLU} \\ \text{Dropout}(0.2) \end{array}\right.$ 1×1, Conv, Compression ratio 0.5 Avg pool, 2×2
3×3, Conv, 512 Batch norm, ReLU 3×3, Conv, 512 Batch norm, ReLU Dropout(0.2) Max pool, 2×1 2×2, Conv, 512 Batch norm, ReLU Dropout(0.2) Max pool, 2×1 Reshape Dense(64)	Dense block 3, 16 layers $\left\{\begin{array}{l} 1 \times 1 \text{ Conv}, 4 \times \text{growth\_rate} \\ \text{Batch norm, ReLU} \\ \text{Dropout}(0.2) \\ 3 \times 3 \text{ Conv, growth\_rate} \\ \text{Batch norm, ReLU} \\ \text{Dropout}(0.2) \end{array}\right.$
Output size: $\lceil \frac{W}{8} \rceil \times 64$	Output size: $\lceil \frac{H}{16} \rceil \times \lceil \frac{W}{16} \rceil \times 684$

7 Results

7.1 Quantitative Results

We can use character and word error rates to quantitatively compare the proposed method with other approaches. The character error rate (CER) between target and output text is calculated using the Levenshtein edit distance formula

$$\mathrm{CER} = \frac{\mathrm{I} + \mathrm{S} + \mathrm{D}}{\mathrm{N}} \times 100 \tag{22}$$

where I, S and D represent the minimum numbers of insertions, substitutions and deletions required to convert the target text string into the output text string and N represents the total number of characters in the target text. To calculate word error rate (WER), Equation (22) can be used after replacing characters with words. To calculate character recognition rate (CRR) and word

Table 6: Architectural details of all decoders used in the experiments. CA = Contextual Attention, CAL = Contextual Attention Localization.

Decoder	**Architecture**
GRU	GRU, 256 GRU, 256 Dropout(0.2)
LSTM	LSTM, 256 LSTM, 256 Dropout(0.2)
BGRU	BGRU $\begin{cases}\text{GRU, 256, Forward}\\\text{GRU, 256, Backward}\end{cases}$ BGRU $\begin{cases}\text{GRU, 256, Forward}\\\text{GRU, 256, Backward}\end{cases}$ Dropout(0.2)
BLSTM	BLSTM $\begin{cases}\text{LSTM, 256, Forward}\\\text{LSTM, 256, Backward}\end{cases}$ BLSTM $\begin{cases}\text{LSTM, 256, Forward}\\\text{LSTM, 256, Backward}\end{cases}$ Dropout(0.2)
CA/CAL	GRU, 256 CA/CAL Unit GRU, 256 Dropout(0.2)

Table 7: Dimensions of learnable matrices and vectors in the proposed decoding framework.

Equation	**Dimensions**
(5), (6), (7)	$k = 130, m = 256, h = 256$
(10)	$f = 11, q = 512$
(11)	$n = 256, d = 684$

recognition rate (WRR), the respective error rates are subtracted from 100[5].

$$\mathrm{CRR} = 100 - \mathrm{CER} \tag{23}$$

$$\mathrm{WRR} = 100 - \mathrm{WER} \tag{24}$$

7.1.1 PUCIT-OHUL dataset

Table 8 shows the comparison of previous models with our proposed model. We compare our model with CNN encoder based uni-directional as well as bidirectional decoders that do not use attention. The CNN architecture that we use has previously been used for handwritten Urdu, Arabic, English and French. The CNN model used in the first 4 models in Table 8 is based on the architecture from [24] and each method was trained on our dataset. Bidirectional decoders were able to improve character recognition rates by less than 2%. Since character recognition rates were so low, word recognition rates are understandably poor. This is why previous work on handwritten Arabic-like scripts does not report word recognition rates. In contrast, our DenseNet encoder jointly trained with a contextual attention based decoder more than doubles both character and word recognition rates. Using the proposed localization penalty on attention further increased character recognition rate by 4.32% and word recognition rate by 6.42%. We believe the reasons for such drastic improvement in results include the following.

1. The DenseNet encoder learns more diverse features compared to the CNN model of previous approaches.
2. Attention reduces the need for bidirectional decoding.
3. Using context, before deciding where to attend, captures the right-to-left, sequential nature of Urdu and Arabic text.
4. Localization penalty encourages the model to attend to specific characters, thereby making the recognition problem less ambiguous.

[5]Theoretically, CER and WER can be greater than 100 but for any reasonably performing recognizer, they are typically less than 100.

Table 8: Comparison of different models for offline handwritten Urdu text recognition on PUCIT-OHUL dataset.

Models	**CRR**	**WRR**
CNN-GRU	33.00	22.44
CNN-LSTM	32.04	21.42
CNN-BGRU	32.76	23.16
CNN-BLSTM [24]	32.02	22.23
SimpleHTR [58]	14.94	5.09
LineHTR [32]	30.05	18.19
CRNN [60]	32.86	24.18
CNN-Contextual Attention	71.42	39.72
DenseNet-Contextual Attention	77.74	45.55
DenseNet-CALText	**82.06**	**51.97**

Table 9 describes the average inference time, memory footprints, and number of parameters for each model. It can be observed that number of parameters in proposed DenseNet-based model are less than other models with CNN-based encoders. This is because DenseNet uses bottleneck layers that employ 1×1 convolutions to compress channel growth as number of layers is increased.

7.1.2 KHATT dataset

We also compare proposed models with the MDLSTM model used for recognition of Arabic offline handwritten KHATT unique lines dataset in Table 10. The state-of-the-art [3] pre-processes the input images by performing skew and slant correction and employs a multi-directional LSTM decoder. Skew and slant correction make the recognition problem significantly easier. Our contextual attention based, directionless decoder without any skew or slant correction almost matches the performance of [3]. Adding the proposed localization penalty further increases character recognition rate by 2.46% and the word recognition rate by 5.41%.

7.2 Qualitative

In order to explain the results of the model, we can visualize the attention weights α_{tl} for region l at time t. Let α_t denote the attention map array for all locations at time t. This attention map can be superimposed on the original image. Figure 10 demonstrates the regions of attention for the proposed CALText model when decoding a sequence of output characters. It can be seen that the model implicitly segments and recognizes each character by focusing only on localized, relevant areas while considering the context of previously attended locations. The purple region indicates the region of attention. It can be observed that the model has learned to attend from right-to-left which is the natural order of Urdu script. The most probable character at each location inferred after beam-search is shown on the right.

Table 9: The average inference time, memory footprint, and number of parameters of different models.

Model	Average Inference time (Sec)	Memory footprint (MB)	Trainable Parameters (Million)
CNN-GRU	0.17	70.9	6.18
CNN-LSTM	0.67	72.8	6.34
CNN-BGRU	0.28	80.4	7.00
CNN-BLSTM [24]	0.31	85.3	7.43
SimpleHTR [58]	0.26	19.1	1.65
LineHTR [32]	1.32	127.8	11.15
CRNN [60]	0.67	77.1	6.71
CNN-Contextual Attention	0.91	114	9.01
DenseNet-Contextual Attention	0.54	114	5.44
DenseNet-CALText	0.54	114	5.44

A more compact visualization can be achieved by exploiting color to represent time. Let the optimal output after beam-search consist of T steps. By dividing time by T we can normalize it so that $0 \leq t \leq 1$. The normalized value of t can be used to compute the color vector $\mathbf{c}_t$ at time t by linearly interpolating between the color vector $\mathbf{c}_0$ at time 0 and $\mathbf{c}_1$ at time 1 as

$$\mathbf{c}_t = (1-t)\mathbf{c}_0 + t\mathbf{c}_1 \tag{25}$$

The final colored attention through time can be visualized as

$$\tilde{\alpha} = \sum_{t \in \{0,\ldots,1\}} \mathbf{c}_t \alpha_t \tag{26}$$

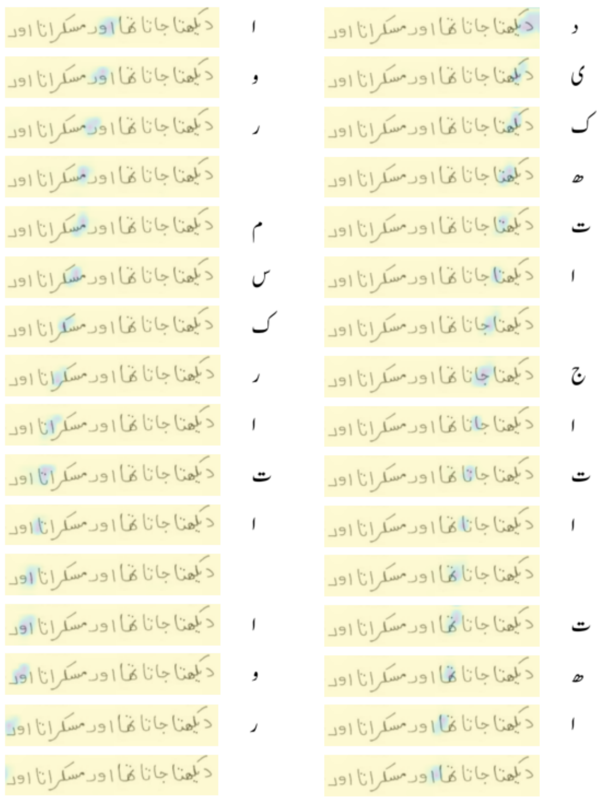
The colored spatiotemporal attention image $\tilde{\alpha}$ visualizes attention both in terms of spatial intensity and temporal sequence. We set $\mathbf{c}_0$ equal to yellow and $\mathbf{c}_1$ equal to green. Therefore, in our visualizations, temporal sequence of attended locations moves from yellow in the beginning to green at the end. All subsequent visualizations (Figures 11 – 15) use this mechanism.

On handwritten Urdu text, Figures 11 and 12 demonstrate three properties of the proposed model.

1. Encouraging the decoder to localize its attention improves both character and word recognition rates.
2. Temporally, learned attention moves from right-to-left which is the natural order of Urdu script.
3. For skewed lines, attention follows the text.

Table 10: Comparison of proposed model with models proposed for recognition of offline handwritten Arabic KHATT dataset. Pre-processing such as skew and slant correction make the recognition problem significantly easier. The proposed CALText model exceeds the state of the art even without any such pre-processing. DSk = Deskewing, DSl = Deslanting, WSP = White space pruning, and IN = Intensity normalization.

Models	**Pre-processing**	**CRR**	**WRR**
HMM [37] (statistical features)	–	46.00	–
HMM [37] (adaptive gradient features)	–	51.20	–
Raw-MDLSTM [3]	DSk+DSl+WSP+IN	75.80	–
CNN-GRU	WSP+IN	36.74	15.66
CNN-LSTM	WSP+IN	39.46	18.77
CNN-BGRU	WSP+IN	39.43	19.53
CNN-BLSTM [24]	WSP+IN	38.40	18.90
SimpleHTR [58]	WSP+IN	29.57	12.15
LineHTR [32]	WSP+IN	38.41	17.06
CRNN [60]	WSP+IN	40.28	21.45
DenseNet-Contextual Attention	WSP+IN	75.01	32.25
DenseNet-CALText	WSP+IN	**77.47**	**37.66**



CRR = 100%, WRR = 100%

Fig. 10: Attention visualization. The proposed CALText model implicitly segments and recognizes every character by focusing only on localized, relevant areas. It also learns to focus in context with attention sequentially moving from right-to-left.

Figure 13 demonstrates the same properties when the CALText model is trained and evaluated for handwritten Arabic text. Figure 14 demonstrates the regions attended by our model when observing an empty white image containing no text. The model trained for right-to-left ground-truth annotations correctly outputs an empty string as the recognition result at the first time step $t = 1$ by focusing on the left-end of the image. In contrast, a model trained on left-to-right ground-truth annotations outputs an empty string after attending the right end of the image. Figure 15 demonstrates that the proposed model has learned to attend directly to text. Instead of blindly scanning the image from left to right, it starts attending the beginning of the sentence text and proceeds until the end of the sentence text, irrespective of the spatial locations of the beginning and end.

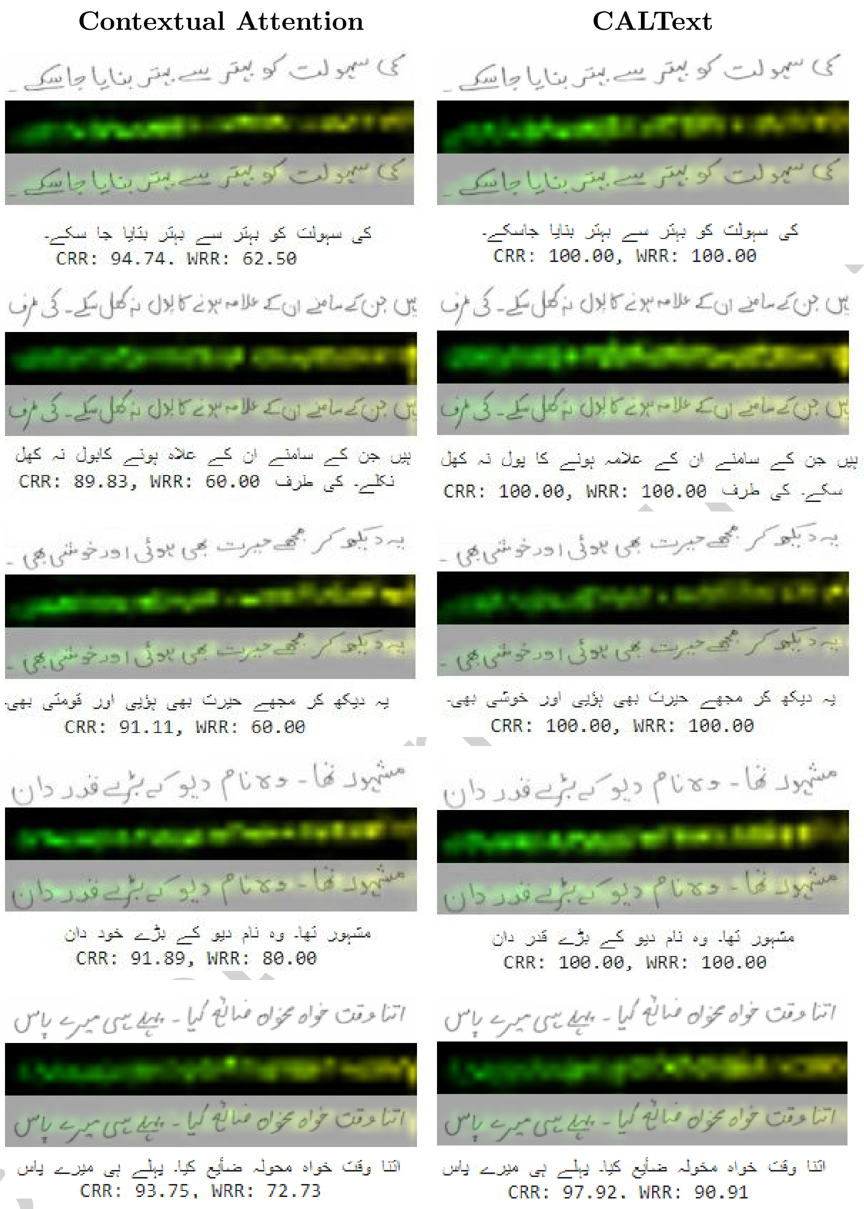
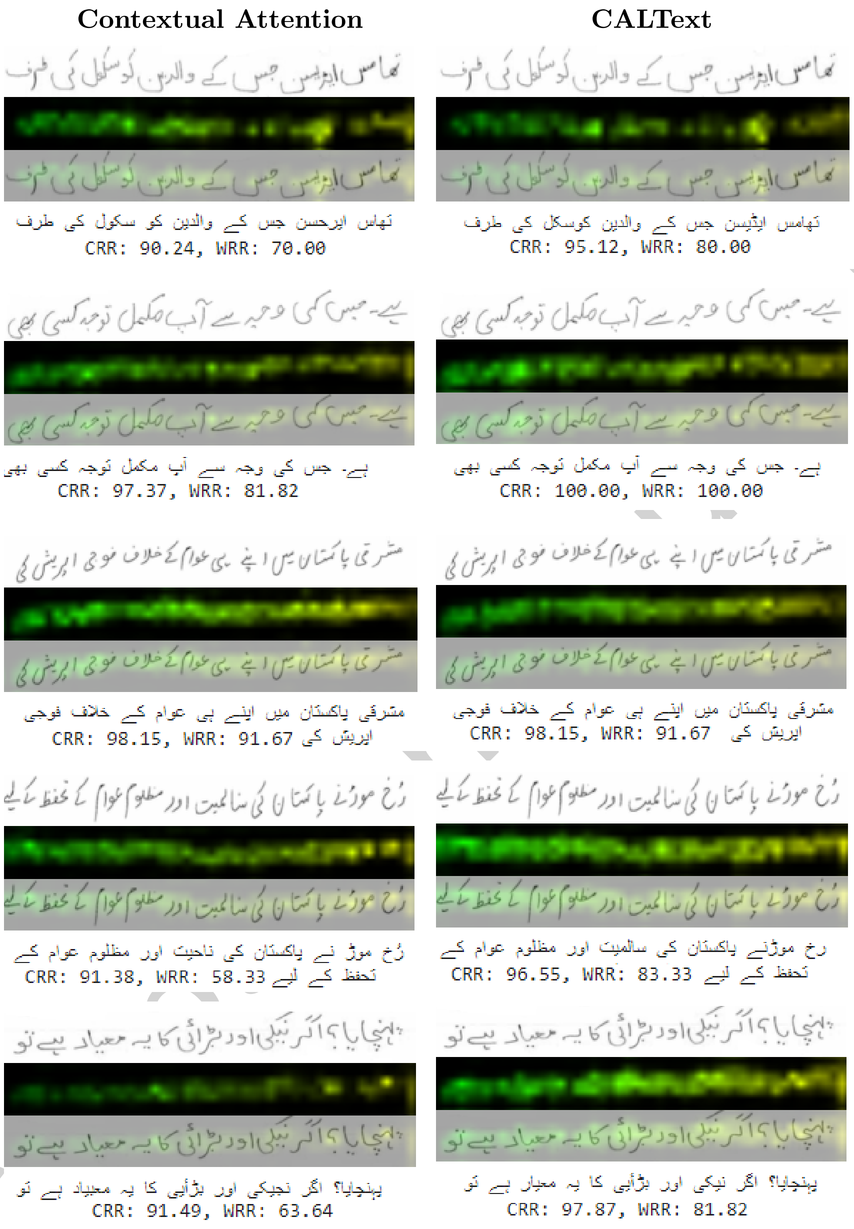

Contextual Attention
CALText
کی سہولت کو بہتر سے بہتر بنایا جا سکے۔
CRR: 94.74. WRR: 62.50
کی سہولت کو بہتر سے بہتر بنایا جاسکے۔
CRR: 100.00, WRR: 100.00
ہیں جن کے سامنے ان کے علاوہ ہونے کابول نہ کھل نکلے۔ کی طرف
CRR: 89.83, WRR: 60.00
ہیں جن کے سامنے ان کے علامہ ہونے کا بول نہ کھل سکے۔ کی طرف
CRR: 100.00, WRR: 100.00
یہ دیکھ کر مجھے حیرت بھی ہوئی اور قومتی بھی۔
CRR: 91.11, WRR: 60.00
یہ دیکھ کر مجھے حیرت بھی ہوئی اور خوشی بھی۔
CRR: 100.00, WRR: 100.00
مشہور تھا۔ وہ نام دیو کے بڑے خود دان
CRR: 91.89, WRR: 80.00
مشہور تھا۔ وہ نام دیو کے بڑے قدر دان
CRR: 100.00, WRR: 100.00
اتنا وقت خواہ مخولہ ضائع کیا۔ پہلے ہی میرے پاس
CRR: 93.75, WRR: 72.73
اتنا وقت خواہ مخولہ ضائع کیا۔ پہلے ہی میرے پاس
CRR: 97.92. WRR: 90.91

Fig. 11: Urdu recognition results on PUCIT-OHUL dataset. In each panel, top row is the input image, second row visualizes spatiotemporal attention, third row superimposes attention over the input, and fourth row is the recognized text string followed by the character and word recognition rates. Transition from yellow to green indicates a temporal sequence of spatial attention.



تهاس ایرحسن جس کے والدین کو سکول کی طرف
CRR: 90.24, WRR: 70.00

تهامس ایڈیسن جس کے والدین کوسکل کی طرف
CRR: 95.12, WRR: 80.00

ہے۔ جس کی وجہ سے آپ مکمل توجہ کسی بھی
CRR: 97.37, WRR: 81.82

ہے۔ جس کی وجہ سے آپ مکمل توجہ کسی بھی
CRR: 100.00, WRR: 100.00

مشرقی پاکستان میں اپنے ہی عوام کے خلاف فوجی
CRR: 98.15, WRR: 91.67 ایریش کی

مشرقی پاکستان میں اپنے ہی عوام کے خلاف فوجی
CRR: 98.15, WRR: 91.67 ایریش کی

رُخ موڑ نے پاکستان کی ناحیت اور مظلوم عوام کے
CRR: 91.38, WRR: 58.33 تحفظ کے لیے

رخ موڑنے پاکستان کی سالمیت اور مظلوم عوام کے
CRR: 96.55, WRR: 83.33 تحفظ کے لیے

پہنچایا؟ اگر نجیکی اور بڑأیی کا یہ معیاد ہے تو
CRR: 91.49, WRR: 63.64

پہنچایا؟ اگر نیکی اور بڑأیی کا یہ معیار ہے تو
CRR: 97.87, WRR: 81.82

Fig. 12: Some more results on handwritten Urdu recognition.

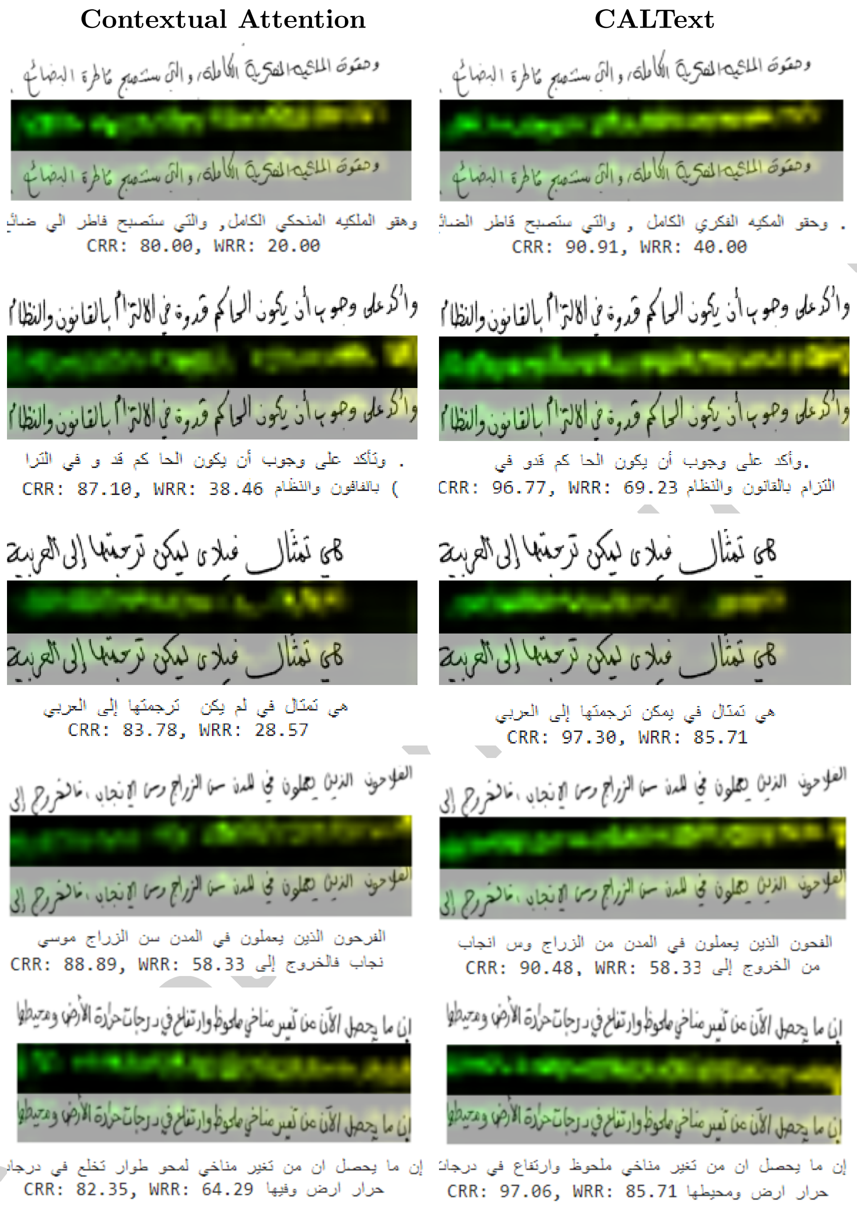
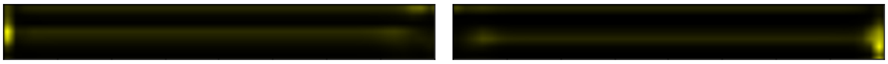

Contextual Attention
CALText
وهقو الملكيه المنحكي الكامل, والتي ستصبح فاطر الي ضائ
CRR: 80.00, WRR: 20.00
. وحقو المكيه الفكري الكامل , والتي ستصبح قاطر الضائ
CRR: 90.91, WRR: 40.00
. وتأكد على وجوب أن يكون الحا كم قد و في الترا
CRR: 87.10, WRR: 38.46 بالقافون والنظام (
وأكد على وجوب أن يكون الحا كم قدو في.
CRR: 96.77, WRR: 69.23 التزام بالقانون والنظام
هي تمثال في لم يكن ترجمتها إلى العربي
CRR: 83.78, WRR: 28.57
هي تمثال في يمكن ترجمتها إلى العربي
CRR: 97.30, WRR: 85.71
الفرحون الذين يعملون في المدن سن الزراج موسي
CRR: 88.89, WRR: 58.33 نجاب فالخروج إلى
الفحون الذين يعملون في المدن من الزراج وس انجاب
CRR: 90.48, WRR: 58.33 من الخروج إلى
إن ما يحصل ان من تغير مناخي لمحو طوار تخلع في درجاد
CRR: 82.35, WRR: 64.29 حرار ارض وفيها
إن ما يحصل ان من تغير مناخي ملحوظ وارتفاع في درجاذ
CRR: 97.06, WRR: 85.71 حرار ارض ومحيطها

Fig. 13: Arabic recognition on KHATT dataset. The proposed CALText model learns to attend to localized regions of the image in context. The transition from yellow to green indicates a temporal sequence of attention. The model learns to attend to Arabic text in a right-to-left direction and follows any skew of the text line.



(a) Model trained for right-to-left recognition. (b) Model trained for left-to-right recognition.

Fig. 14: Behavior of the proposed CALText model on an empty white image. Attention visualization for a model trained on (a) right-to-left text, and (b) left-to-right text. Both models output an empty string. As indicated by the yellow color, the outputs were obtained at the very first time step $t = 1$. It can be noted that before outputting an empty string, the models focus more on the relevant end of the image. However, they do focus, albeit less, on the whole horizontal extant of the image as well. This is intuitively satisfying since the whole image *should* be scanned before outputting an empty string.

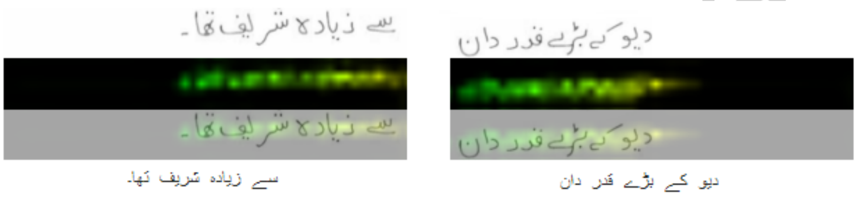


Fig. 15: In order to produce its sequence of output characters, the proposed CALText model learns to attend to text regions exclusively.

8 Conclusion

We have proposed an encoder-decoder recognition of offline, handwritten text. The model has been evaluated for Urdu and Arabic which share numerous common features such as right-to-left ordering, similar characters and two-dimensional writing structure with inconsistent overlaps and spacing. Multiscale features are extracted through a DenseNet encoder. To incorporate the role of context in the process of *reading* and to attend to specific characters, we have introduced a contextual attention localizer between two gated recurrent units. Results on test images show that our proposed CALText model learns to read and recognize text in a way that is similar to human reading by focusing only on relevant and localized image regions. The model determines relevancy of a region through contextual attention.

We have comprehensively re-annotated all the ground-truth text lines of the PUCIT-OHUL dataset. As a result of our re-annotation, the count of unique characters increased from 98 to 130. Numerous mistakes and omissions in the original annotations have been corrected. For Urdu text, the use of contextual attention improves upon the state-of-the-art by more than a factor of 2× in terms of both character and word recognition rates. Encouraging the attention to be localized increased character recognition rate by 4.32% and word recognition rate by 6.42% for Urdu. The corresponding increases for

Arabic were 2.46% and 5.41%. Code and pre-trained models are made publicly available at https://github.com/nazar-khan/CALText.

In the proposed model, attention has been exploited only in the decoder. One potential improvement could involve attention-based encodings as well [19]. Two-stage training using anomalous samples [73] and data augmentation at both training as well as testing stages [72] to let the model behave as an implicit ensemble during training as well as testing are interesting future directions for better generalization.

Acknowledgments

This research was supported by HEC-NRPU Grant 8329. Clean annotations of the PUCIT-OHUL dataset were assisted by Adeela Islam, Asmat Batool, Rabia Sirhindi, Tauseef Iftikhar and Abubakar Siddique.

Conflict of Interest

The authors have no conflicts of interest (financial or otherwise).

References

[1] What are the top 200 most spoken languages? https://www.ethnologue.com/guides/ethnologue200, 2021. accessed: 18.09.2021.

[2] Haikal El Abed and Volker Margner. The IFN/ENIT-database - a tool to develop Arabic handwriting recognition systems. *Int'l Journal on Document Analysis and Recognition*, 5(10):39 – 46, 2002.

[3] Riaz Ahmad, Saeeda Naz, M. Zeshan Afzal, S. Faisal Rashid, Marcus Liwicki, and Andreas Dengel. KHATT: A deep learning benchmark on Arabic script. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017.

[4] Saad Bin Ahmed, Saeeda Naz, Salahuddin Swati, and Muhammad Imran Razzak. Handwritten Urdu character recognition using one-dimensional BLSTM classifier. *Neural Computing and Applications*, 31(4):1143–1151, April 2019.

[5] Q. A. Akram, S. Hussain, A. Niazi, U. Anjum, and F. Irfan. Adapting Tesseract for complex scripts: An example for Urdu Nastalique. In *Proceedings of 11th IAPR Workshop on Document Analysis Systems (DAS 14)*, 2014.

[6] Huda Alamri, Javad Sadri, Ching Y. Suen, and Nicola Nobile. A novel comprehensive database for Arabic offline handwriting Recognition . *Journal Pattern Recognition*, 60:378–393, 2016.

[7] Najoua Ben Amara, Omar Mazhoud, Noura Bouzrara, and Noureddine Ellouze. ARABASE: A relational database for Arabic OCR systems. *International Arab Journal of Information Technology*, 2(4):259–266, 2005.

[8] T. Anjum and N. Khan. An attention based method for offline handwritten Urdu text recognition. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 169–174, 2020.

[9] J. Aradillas, J. Murillo-Fuentes, and P. M. Olmos. Improving offline HTR in small datasets by purging unreliable labels. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 25–30, Los Alamitos, CA, USA, Sep 2020. IEEE Computer Society.

[10] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio. End-to-end attention-based large vocabulary speech recognition. *Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949, 2016.

[11] A. Baro, C. Badal, and A. Fornes. Handwritten historical music recognition by sequence-to-sequence with attention mechanism. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 205–210, Los Alamitos, CA, USA, Sep 2020. IEEE Computer Society.

[12] Muhammad Farrukh Bashir, Abdul Rehman Javed, Muhammad Umair Arshad, Thippa Reddy Gadekallu, Waseem Shahzad, and Mirza Omer Beg. Context aware emotion detection from low resource urdu language using deep neural network. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, Mar 2022.

[13] S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri, and D. K. Basu. A novel framework for automatic sorting of postal documents with multi-script address blocks. *Pattern Recognition*, 43(10):3507–3521, 2010.

[14] Ali Borji, Mandana Hamidi, and Fariborz Mahmoudi. Robust handwritten character recognition with features inspired by visual ventral stream. *Neural Processing Letters*, 28(2):97–111, 2008.

[15] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *ArXiv e-prints*, 2014.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint*, 2014.

[17] D. Coquenet, C. Chatelain, and T. Paquet. Recurrence-free unconstrained handwritten text recognition using gated fully convolutional network. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 19–24, Los Alamitos, CA, USA, Sep 2020. IEEE Computer Society.

[18] Nibaran Das, Jagan Mohan Reddy, Ram Sarkar, Subhadip Basu, Mahantapas Kundu, Mita Nasipuri, and Dipak Kumar Basu. A statistical–topological feature combination for recognition of handwritten numerals. *Applied Soft Computing*, 12(8):2486–2495, 2012.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[20] Bernardo B Gatto, Eulanda M dos Santos, Kazuhiro Fukui, Waldir SS Júnior, and Kenny V dos Santos. Fukunaga–Koontz convolutional network with applications on character classification. *Neural Processing Letters*, 52(1):443–465, 2020.

[21] Bernard Gosselin. Multilayer perceptrons combination applied to handwritten character recognition. *Neural Processing Letters*, 3(1):3–10, 1996.

[22] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. *In Advances in Neural Information Processing Systems (NIPS)*, 21:5454–552, 2009.

[23] Gui, Liangke, Xiaodan Liang, Xiaojun Chang, and Alexander G Hauptmann. Adaptive context-aware reinforced agent for handwritten text recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[24] S. Hassan, A. Irfan, A. Mirza, and I. Siddiqi. Cursive handwritten text recognition using bi-directional LSTMs: A case study on Urdu handwriting. In *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*, pages 67–72, 2019.

[25] Abhishek Hazra, Prakash Choudhary, Sanasam Inunganbi, and Mainak Adhikari. Bangla-meitei mayek scripts handwritten character recognition using convolutional neural network. *Applied Intelligence*, 51(4):2291–2311, 2021.

[26] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[27] Mujtaba Husnain, Malik Muhammad Saad Missen, Shahzad Mumtaz, Mickaël Coustaty, Muzzamil Luqman, and Jean-Marc Ogier. Urdu handwritten text recognition: a survey. *IET Image Processing*, 14(11):2291–2300, 2020.

[28] S. T. Javed and S. Hussain. Improving Nastalique specific pre recognition process for Urdu OCR. In *Proceedings of 13th IEEE International Multitopic Conference (INMIC)*, 2009.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[30] Munish Kumar, Simpel Rani Jindal, Manish Kumar Jindal, and Gurpreet Singh Lehal. Improved recognition results of medieval handwritten gurmukhi manuscripts using boosting and bagging methodologies. *Neural Processing Letters*, 50(1):43–56, 2019.

[31] Hicham Lamtougui, Hicham El Moubtahij, Hassan Fouadi, Ali Yahyaouy, and Khalid Satori. Offline arabic handwriting recognition using deep learning: Comparative study. In *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–8, 2020.

[32] HT Lcm. Line-level handwritten text recognition with tensorflow. https://github.com/lamhoangtung/ LineHTR, 2018. accessed: 20.07.2022.

[33] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998. accessed: 19.07.2019.

[34] Z. Li, L. Jin, S. Lai, and Y. Zhu. Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 175–180, Los Alamitos, CA, USA, Sep 2020. IEEE Computer Society.

[35] N. Ly, C. Nguyen, and M. Nakagawa. Attention augmented convolutional recurrent network for handwritten japanese text recognition. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 163–168, Los Alamitos, CA, USA, Sep 2020. IEEE

Computer Society.

[36] N.T. Ly, H.T. Nguyen, and M. Nakagawa. 2D self-attention convolutional recurrent network for offline handwritten text recognition. In *2021 International Conference on Document Analysis and Recognition (ICDAR)*, 2021.

[37] Sabri A. Mahmoud, Irfan Ahmad, Wasfi G. Al-Khatib, Mohammad Alshayeb, Mohammad Tanvir Parvez, Volker Margner, and Gernot A. Fink. KHATT: An open Arabic offline handwritten text database. *Pattern Recognition*, 47(3):1096–1112, 2014.

[38] Sabri A Mahmoud and Wasfi G Al-Khatib. Recognition of Arabic (Indian) bank check digits using log-gabor filters. *Applied Intelligence*, 35(3):445–456, 2011.

[39] U. Marti and H. Bunke. The IAM-database: An English sentence database for off-line handwriting recognition . *Int'l Journal on Document Analysis and Recognition*, 5(10):39 – 46, 2002.

[40] H. Mouchere, C. Viard-Gaudin, R. Zanibi, D. H. Kim, J. H. Kim, and U. Garain. ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions. In *Proc. ICDAR*, 2013.

[41] Omar Mukhtar, Srirangaraj Setlur, and Venu Govindaraju. *Experiments on Urdu Text Recognition*, pages 163–171. Springer London, London, 2010.

[42] Tayyab Nasir, Muhammad Kamran Malik, and Khurram Shahzad. MMU-OCR-21: Towards End-to-End Urdu Text Recognition Using Deep Learning. *IEEE Access*, 9:124945–124962, 2021.

[43] S. Naz, S.B. Ahmed, R. Ahmad, and M.I. Razzak . Zoning features and 2D LSTM for Urdu text-line recognition. In *Procedia Comput Sci*, pages 16–22, 2016.

[44] S. Naz, A.I. Umar, R. Ahmad, S.B. Ahmed, S.H. Shirazi, and M.I. Razzak. Urdu Nastaliq text recognition system based on multi-dimensional recurrent neural network and statistical features. *Neural Computing and Applications*, pages 219–231, 2017.

[45] S. Naz, A.I. Umar, R. Ahmad, I. Siddiqi, S.B. Ahmed, M.I. Razzak, and F. Shafait. Urdu Nastaliq recognition using convolutional recursive deep learning. *Neurocomputing*, pages 80–27, 2017.

[46] Saeeda Naz, Arif I. Umar, Syed H. Shirazi, Saad B. Ahmed, Muhammad I. Razzak, and Imran Siddiqi. Segmentation techniques for recognition of

Arabic-like scripts: A comprehensive survey. *Educ Inf Technol*, 20(1), 2015.

[47] Salima Nebti and Abdellah Boukerram. Handwritten characters recognition based on nature-inspired computing and neuro-evolution. *Applied Intelligence*, 38(2):146–159, 2013.

[48] K. Nguyen, C. Nguyen, and M. Nakagawa. A semantic segmentation-based method for handwritten Japanese text recognition. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 127–132, Los Alamitos, CA, USA, Sep 2020. IEEE Computer Society.

[49] Mohammad Tanvir Parvez and Sabri A. Mahmoud. Offline arabic handwritten text recognition: A survey. *ACM Comput. Surv.*, 45(2), Mar 2013.

[50] I.K. Pathan and A. Ali. Recognition of offline handwritten isolated Urdu character. *Advances in Computational Research*, 4(1):117–121, 2012.

[51] Marcus Pfister, Sven Behnke, and Raúl Rojas. Recognition of handwritten zip codes in a real-world non-standard-letter sorting system. *Applied Intelligence*, 12(1):95–115, 2000.

[52] A. Raza, I. Siddiqi, A. Abidi, and F. Arif. An unconstrained benchmark Urdu handwritten sentence database with automatic line segmentation. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 491–496, 2012.

[53] Javad Sadri, Mohammad Reza Yeganehzad, and Javad Saghi. A novel comprehensive database for offline Persian handwriting recognition. *Pattern Recognition*, 60:378–393, 2016.

[54] Khalid Saeed and Majida Albakoor. Region growing based segmentation algorithm for typewritten and handwritten text recognition. *Applied Soft Computing*, 9(2):608–617, 2009.

[55] M. W. Sagheer, C. L. He, N. Nobile, and C. Y. Suen. Holistic Urdu handwritten word recognition using support vector machine. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR 10)*, pages 1900–1903, 2010.

[56] Malik Waqas Sagheer, Chun Lei He, Nicola Nobile, and Ching Y Suen. A new large Urdu database for off-line handwriting recognition. In *International Conference on Image Analysis and Processing*, pages 538–546. Springer, 2009.

[57] Kenneth M Sayre. Machine recognition of handwritten words: A project report. *Pattern recognition*, 5(3):213–228, 1973.

[58] H Scheidl. Handwritten text recognition with tensorflow. https://github.com/githubharald/ SimpleHTR, 2018. accessed: 20.07.2022.

[59] Shibaprasad Sen, Mridul Mitra, Ankan Bhattacharyya, Ram Sarkar, Friedhelm Schwenker, and Kaushik Roy. Feature selection for recognition of online handwritten Bangla characters. *Neural Processing Letters*, 50(3):2281–2304, 2019.

[60] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[61] Kazem Taghva, Thomas A. Nartker, Julie Borsack, and Allen Condit. UNLV-ISRI document collection for research in OCR and information retrieval. In *Proc. SPIE 3967, Document Recognition and Retrieval VII*, 1999.

[62] A. Tong, M. Przybocki, V. Margner, and H. El Abed. NIST 2013 open handwriting recognition and translation evaluation. In *Proc. NIST*, 2013.

[63] A. Ul-Hasan, S.B. Ahmed, F. Rashid, F. Shafait, and T.M. Breuel. Offline printed Urdu Nastaleeq script recognition with bidirectional LSTM networks. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1061–1065. IEEE, 2013.

[64] A. Ul-Hasan, F. Shafait, and M. Liwicki. Curriculum learning for printed text line recognition of ligature-based scripts. In *Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1001–1005. IEEE, 2015.

[65] Kaili Wang, Yaohua Yi, Ziwei Tang, and Jibing Peng. Multi-scene ancient chinese text recognition with deep coupled alignments. *Applied Soft Computing*, 108:107475, 2021.

[66] Ronald J. Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 06 1989.

[67] Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. *CoRR*, abs/1606.02960, 2016.

[68] Yao Xiao, Dan Meng, Cewu Lu, and Chi-Keung Tang. Template-instance loss for offline handwritten chinese character recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 315–322, 2019.

[69] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701, 2012.

[70] Jianshu Zhang, Jun Du, and Lirong Dai. Multi-scale attention with dense encoder for handwritten mathematical expression recognition. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2245–2250. IEEE, 2018.

[71] Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jinshui Hu, Si Wei, and Lirong Dai. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71:196–206, 2017.

[72] Qinghe Zheng, Mingqiang Yang, Xinyu Tian, Nan Jiang, and Deqiang Wang. A full stage data augmentation method in deep convolutional neural network for natural image classification. *Discrete Dynamics in Nature and Society*, 2020, 2020.

[73] Qinghe Zheng, Mingqiang Yang, Jiajie Yang, Qingrui Zhang, and Xinxin Zhang. Improvement of generalization ability of deep CNN via implicit regularization in two-stage training process. *IEEE Access*, 6:15844–15869, 2018.